PartyPal: aplicació *matchmaking* orientada a esdeveniments

Memòria del Projecte

Treball Final de Grau

Autor:

Joan Sánchez García

Grau en Enginyeria Informàtica

Especialitat en Enginyeria del Software

Directora:

Cristina Gómez Seoane

Departament d'Enginyeria de Serveis

i Sistemes d'Informació.

5 de Juliol 2019

Agraïments

A la meva tutora, la Cristina Gómez Seoane, per haver-me fet estimar el software i haver-me donat la llibertat i confiança que necessitava per dur a terme aquest projecte.

A la meva família, la Julia, en Francisco i la Maria, per no haver dubtat de mi en cap moment.

I a tota la gent que m'he trobat en aquest bonic camí de 4 anys que, de forma desinteressada, han estat allà per recolzar-me i ajudar-me a arribar a estar entregant aquest treball.

RESUM

En la societat actual, seguint la tradició des de les èpoques dels primers homínids, reunir-se amb els col·lectius als quals pertanyem segueix tenint un pes molt important. En l'actualitat sorgeix el concepte "d'esdeveniment", que engloba fets importants dins d'una comunitat.

En la recerca de descobrir nous mètodes de socialització, a mitjans de la primera dècada dels 2000 apareixen les primeres aplicacions matchmaking. Aquestes consisteixen en la recomanació consecutiva de perfils d'altres usuaris del sistema, que l'usuari pot acceptar o rebutjar. En cas que els dos usuaris s'acceptin mútuament es genera match i es dona accés a recursos compartits.

El sistema que s'ha de desenvolupar busca oferir un punt d'unió en forma de xat entre persones que assistiran a un mateix esdeveniment abans de la seva celebració. L'aplicació oferirà l'opció de seguir els esdeveniments en els quals es pugui estar interessat i recomanarà usuaris que també hi assistiran; permet, per tant, la socialització selectiva: hi ha un tema comú.

RESUMEN

En la sociedad actual, siguiendo la tradición desde las épocas de los primeros homínidos, reunirse con los colectivos a los que pertenecemos sigue teniendo un peso muy importante. En la actualidad surge el concepto de "evento", que engloba hechos importantes dentro de una comunidad.

En la búsqueda de descubrir nuevos métodos de socialización, a mediados de la primera década de los 2000 aparecen las primeras aplicaciones matchmaking. Estas consisten en la recomendación consecutiva de perfiles de otros usuarios del sistema, que el usuario puede aceptar o rechazar. En caso de que los dos usuarios se acepten mutuamente se genera match y se da acceso a recursos compartidos.

El sistema que se ha de desarrollar busca ofrecer un punto de unión en forma de chat entre personas que asistirán a un mismo evento antes de su celebración. La aplicación ofrecerá la opción de seguir los eventos en los que se pueda estar interesado y recomendará usuarios que también asistirán; permite, por tanto, la socialización selectiva: hay un tema común.

ABSTRACT

In today's society, following the tradition from the time of the first hominids, meeting with the groups to which we belong continues to play a very important role. Nowadays, the idea of "event" arises, which encompasses important facts within a community.

In the search to discover new methods of socialization, in the middle of the first decade of 2000, the first matchmaking applications appeared. They consist in the consecutive recommendation of profiles of other users of the system, which the user can accept or reject. In case the two users accept each other, a match is generated and access to shared resources is given.

The system that is to be developed seeks to offer a point of union using a chat format between people who will attend the same event before its celebration. The application will offer the option to follow the events in which you may be interested, plus it will recommend users who will also attend; it allows, therefore, selective socialization: there is a common theme.

Contingut de la memòria

1. Introducció

1.1. Descripció del problema

"Ningún hombre es una isla."
Paulo Coelho

"Individually, we are a drop. Together we are an ocean"
Ryunosuke Satoro

Vivim en una societat cada cop més unida pel que fa a infraestructures i a la vegada més distant en relacions. Si mirem al nostre voltant, veurem gent, molta gent, tancada en les seves bombolles d'aïllament, ja sigui amb un llibre, música, el mòbil o simplement amb els seus pensaments. Veiem cossos que van d'un costat a l'altre sense que sembli que els importi molt allò que es troba a més de mig metre d'ells. Sembla que l'ésser humà s'estigui oblidant de la qualitat que l'ha portat a ser l'espècie dominant en aquest planeta: el poder del grup.

Tot i que una persona per si sola pot arribar a fer coses extraordinàries, no és fins que una col·lectivitat dona suport a una idea o projecte que aquests es fan realitat. Exemples podrien ser la revolució francesa, la revolució cubana, la primavera àrab, la lluita contra la segregació i la discriminació racial als Estats Units de Martin Luther King, la lluita per la igualtat de gènere… Per tant, encara que ens agradi estar sols, al llarg de la història hem demostrat que apel·lem al poder del grup quan el problema que tenim davant supera les nostres capacitats individuals.

La millor manera que hem tingut sempre de socialitzar-nos ha estat a partir de reunions, celebracions, rituals…, que amb el pas del temps hem anat ampliant a concerts, trobades, congressos o manifestacions; el que avui dia podríem generalitzar sota el concepte "d'esdeveniments". Tal com ho defineix el DIEC (Diccionari de l'Institut d'Estudis Catalans), un esdeveniment seria un "Fet extraordinari o important en la vida d'una persona, d'una col·lectivitat social, de la humanitat.". Per tant, fins i tot al llenguatge hem plasmat la importància que donem a la unió com a éssers humans. Com deia Aristòtil: "L'home és un ésser social per naturalesa".

Però a temps moderns, problemes moderns. Quantes vegades ens ha passat que volem assistir a un concert o festival però cap dels nostres amics o coneguts hi va? Hi anem sols o no hi anem? O volem anar-hi i no sabem com? O ens sobren places al cotxe? O quants de nosaltres posem com a excusa sortir de festa per a conèixer gent nova?

Si ens posem en la pell de qualsevol que estigui visquent alguna de les situacions anteriors, veurem que les possibilitats per a poder satisfer les seves necessitats es troben disperses en moltes plataformes i aplicacions diferents: *Facebook, Instagram, BlaBlaCar, Tinder*…

Com més aplicacions hi ha al mercat, més ens costa trobar aquella que unifiqui totes les funcionalitats que busquem per a una tasca determinada. Normalment acabem retallant les nostres expectatives o descarregant més d'una aplicació per a poder satisfer-les.

El sistema que s'ha de desenvolupar busca oferir un punt d'unió en forma de xat entre persones que assistiran a un mateix esdeveniment abans de la seva celebració. L'aplicació oferirà l'opció de seguir els esdeveniments en els quals es pugui estar interessat i recomanarà usuaris que també hi assistiran; permet, per tant, la socialització selectiva: hi ha un tema comú.

1.2. Motivació

Des de ben petit sempre he sentit especial inclinació per la part pràctica del món més que per la teòrica. Amb el temps, i no sense gran esforç del pare, de la mare i del professorat, vaig anar entenent la importància d'entendre com funcionen les coses abans de posar-li les mans a sobre i, fins i tot, he arribat a sentir-me còmode en l'àmbit teòric. Durant la carrera, és cert que hem tingut aplicacions pràctiques, però sempre limitades a un temps, abast i complexitat finit per a poder-se encabir dins de l'ajustat pla docent. Com amb el nen petit quan es va fent gran, que se li dona més responsabilitat, aquest cop i per aquest projecte, les expectatives són diferents i l'abast molt més ambiciós.

Aquest treball es presenta com un repte personal: tinc l'oportunitat de dur a la pràctica 4 anys de coneixements fent realitat una d'aquelles idees escrites a la sovint ajornada llista de "projectes per fer". El problema que ens ocupa comprèn diversos camps, tant socials com tecnològics, que m'interessen molt.

Per una part, em sembla sorprenent que un algoritme pugui decidir a qui coneixes i a qui no. Si es pensa fredament, estem delegant en línies de codi la responsabilitat de filtrar i decidir amb qui creiem que serem afins, sense tenir la més mínima idea dels criteris en què es basa aquesta decisió.

Per altra part, molts dels apartats que conformen l'àmbit tecnològic d'aquest projecte no em són desconeguts, però sí distants. Entenc la imatge general, però em falta aprendre com funciona cada apartat en detall. Ara és l'oportunitat perfecta per a trencar la bombolla de confort i posar-me mans a l'obra per adquirir el bagatge suficient amb aquestes eines per a sentir-me còmode. A més, gràcies als companys que m'han assessorat en quines tecnologies usar, es faran servir tecnologies punteres al mercat, que em seran útils més enllà d'aquest projecte.

Orientar-lo al món dels esdeveniments sorgeix de trobar-me el problema que descric en primera persona. Si la solució a la qual es pugui arribar em pot ajudar a mi, segurament podrà ajudar a algun altre.

1.3. Contingut de la memòria

Aquesta memòria és un recull de tots els passos per a desenvolupar un projecte d'enginyeria del software. Els següents apartats mostren els artefactes amb què treballarien caps de projecte, analistes, dissenyadors, programadors i testers, que en un entorn real, estarien repartits en diverses persones o unitats.

Es començarà estudiant el context i l'abast del projecte i es definirà la metodologia i el rigor que es farà servir. A continuació es realitzarà una anàlisi de requisits per a definir els requeriments funcionals i els no funcionals. Un cop es tingui clar quines funcionalitats hauria d'incloure el sistema, es procedirà a especificar-les formalment. Posteriorment, es realitzarà un disseny preliminar de l'arquitectura del sistema, que s'anirà completant de manera progressiva a mesura que aquest vagi avançant, igual que l'apartat d'implementació. Per acabar, es validarà si s'ha aconseguit satisfer els requisits inicials i s'estudiarà la sostenibilitat i la gestió del projecte.

2. Context

2.1. Conceptes previs

El sistema esmentat consistirà en un servidor Back End[1] i una aplicació Android.

L'aplicació oferirà tres funcionalitats clau: el cercador d'esdeveniments, el sistema de recomanació d'usuaris i el sistema de xat.

Cercador d'esdeveniments

Un cop l'usuari hagi registrat el compte i ja estigui dins de l'aplicació, el primer pas per a interactuar amb altres usuaris serà començar a seguir esdeveniments que li interessin. El sistema oferirà un cercador que funcionarà sobre l'API (Application programming interface) de *Ticketmaster*, per tant es tindrà accés a tots els concerts, teatres, festivals, esports, etc., que aquest servei ofereix.

Un cop l'usuari hagi trobat un esdeveniment que li interessi, podrà seguir-lo. Aquesta acció es tradueix en el fet que vol conèixer gent que també hi assistirà al mateix lloc i, per tant, el sistema de recomanació d'usuaris li mostrà perfils de gent amb la qual té, com a mínim, un esdeveniment en comú.

Els esdeveniments funcionen com baguls d'usuaris als quals el sistema de recomanació pot recórrer per tal d'oferir millors suggerències. Quants més se segueixin, més diverses seran les recomanacions.

L'usuari pot optar per deixar de seguir un esdeveniment i, per tant, deixar de rebre recomanacions d'usuaris que hi assistiran.

Sistema de recomanació d'usuaris

El sistema de recomanació d'usuaris anirà suggerint perfils d'usuaris que segueixen, com a mínim, un esdeveniment en comú amb nosaltres. Funcionarà semblant a *Tinder* o *OKCupid*, aplicacions populars en el món de cites online, pel que fa al fet que les accions respecte a altres usuaris estaran lligades a les interaccions o gestos amb els perfils que es van mostrant.

Per a l'explicació del funcionament es faran servir dos usuaris: a1 i a2. L'usuari a1 estarà veient els perfils que el sistema li recomana. Aquests apareixen un rere l'altre i només es pot accedir al següent un cop s'ha pres una acció en l'actual. Les accions que es podran fer i una possibilitat d'interacció podrien ser les següents:

- Rebutjar usuari a2: usuari a1 llisca el dit de dreta a esquerra indicant al sistema que no li agrada el perfil que li ha mostrat i que li mostri el següent.

[1] Back end: capa de processament de les dades, en terminologia informàtica. Gràficament, conjunt d'engranatges que queden sota la carcassa exterior del rellotge fent que aquest funcioni.

- Enviar missatge a a2: el que es tradueix en llançar petició de match[2]. L'usuari a1 està fent servir el sistema de recomanació i aquest li recomana a2. Com li agrada el perfil d'a2, li vindria de gust tenir match amb ell. La forma més fàcil d'aconseguir-ho és enviant-li un missatge i que a2 respongui. Per a enviar un missatge, a1 llisca el dit d'esquerra a dreta i escriu el text. Un cop a1 l'envia, a2 pot decidir si vol respondre. En cas que ho faci, es crea match i els dos usuaris podran xatejar tant com vulguin. En cas que rebutgi el missatge (hi haurà una opció), no hi haurà match.

- Guardar usuari a2: el que es tradueix en llançar petició de match sense enviar missatge. Si quan el sistema de recomanació recomana a2 a a1 i a aquest li agrada, però no vol enviar-li cap missatge, premerà sobre el botó corresponent per a guardar el perfil. A2 no rebrà cap notificació de nou missatge ni de que ha estat guardat per a1, però el seu sistema de recomanació, en un moment o un altre, li recomanarà a1. Si a2 guarda o envia un missatge a a1, es crearà match i podran xatejar tant com vulguin. Si a2 rebutgés a a1, no hi hauria match.

A part d'aquestes accions, l'usuari a1 també podrà navegar pel perfil i les fotografies de l'usuari a2 abans de prendre la decisió.

Sistema de xat

A part del missatge inicial fent servir el sistema de recomanació, dos usuaris només podran xatejar si són match. El xat seguirà l'estàndard actual: sistema de comunicació 1 a 1 que permetrà intercanviar missatges de text.

El sistema també inclourà l'opció de desfer el match amb l'usuari en qüestió, o de reportar-lo[3] si creiem que el seu comportament no està sent l'adequat. Un usuari que acumuli més de 3 reports serà eliminat del sistema.

2.2. Parts interessades

Suzanne i James Robertson [1] defineixen els actors principals, *stakeholders* en anglès, com "aquella gent que té un interès en l'èxit del producte. Aquest grup inclou a qualsevol que tingui alguna influència sobre el resultat o tingui coneixements necessaris per descobrir els requisits del producte".

Es descriuen a continuació els actors principals del nostre sistema.

[2] match: fet d'agradar o seguir un usuari i que aquest ens correspongui, en xarxes socials.

[3] report: acte de fer una queixa a una persona autoritzada sobre algú o alguna cosa, principalment en el món online.

Usuaris:

Grup que farà ús de forma més directa de les funcionalitats oferides a l'aplicació. Potencialment seran usuaris que estiguin interessats en algun esdeveniment i que comptin amb un dispositiu amb sistema operatiu Android.

Objectius:

- Conèixer gent abans d'anar a un esdeveniment.
- Xatejar amb gent amb els mateixos gustos que tu.
- Assabentar-se de nous esdeveniments.

Rols:

- Seguir o deixar de seguir esdeveniments.
- Reportar usuaris que no estiguin fent un ús adequat del sistema.

Creadors i patrocinadors de l'esdeveniment:

La intenció d'aquest grup és veure augmentar el nombre d'assistents als seus esdeveniments. Ara comptaran amb una altra via de publicitat gratuïta: el boca a boca de la gent dins l'aplicació.

Objectius:

- Augmentar el nombre d'assistents als seus esdeveniments.

Rols:

- Penjar la informació del seu esdeveniment a *Ticketmaster* i mantenir-la actualitzada.

Ticketmaster:

Grup clau dins del sistema. Aporta l'API per a cercar esdeveniments i que els usuaris els puguin seguir. A canvi, tenen una altra via de publicitat per a les seves entrades, donat que l'aplicació indicarà que el motor de cerca d'esdeveniments és d'aquesta plataforma.

Objectius:

- Augmentar el nombre d'entrades venudes als seus esdeveniments.

Rols:

- Mantenir l'API pública.
- Demanar als creadors d'esdeveniments que els mantinguin actualitzats.

Directora del projecte:

El professorat que dirigeix el TFG també té interès en la seva correcta elaboració i finalització. En el cas d'aquest projecte, la Cristina Gómez Seoane, sotsdirectora del departament d'Enginyeria de Serveis i Sistemes d'Informació.

Objectius:

- Assegurar que es compleixen els objectius definits durant la inscripció del projecte.
- Assegurar un ús correcte dels coneixements i eines apresos durant el grau.
- Vetllar perquè el treball pugui lluitar per la màxima nota possible.

Rols:

- Guiar en la correcta elaboració del projecte complint amb uns mínims de qualitat.
- Supervisar els diferents punts del treball abans que siguin avaluats.

Desenvolupador:

Persona a càrrec del desenvolupament d'aquest projecte. Atès que és un treball individual, una mateixa persona concentra tots els rols: dissenyador, programador, enginyer de requisits…

Objectius:

- Aplicar tot el coneixement incorporat durant el transcurs del grau.
- Demostrar expertesa en les competències esperades d'un Enginyer Informàtic de la FIB (Facultat d'Informàtica de Barcelona).
- Assimilar tot el coneixement nou que calgui per a la correcta i completa realització d'aquest projecte.
- Lluitar per la màxima nota possible.

Rols:

- Elaborar, complint amb els màxims estàndards de qualitat possible, la memòria i l'aplicació que conformen aquest projecte.
- Assimilar les recomanacions que pugui rebre per part de la seva directora i introduir-les en el projecte.

2.3. Estat de l'art

Abans de començar cap projecte és de vital importància conèixer com està el mercat en aquest camp, quina podria ser la competència i quines funcionalitats diferencials podem oferir nosaltres. Això és el què s'anomena estat de l'art. S'estudiarà la competència directa, que són aquells resultats que les botigues d'aplicacions ens mostren quan busquem paraules relacionades amb l'aplicació que volem elaborar. S'estudiarà productes relacionats ja establerts al mercat i també als proveïdors d'entrades que farem servir.

Competència directa

- ***PartyWith***[2]: anteriorment *Party With a Local*, aplicació orientada únicament a sortir de festa i conèixer gent nova. No ofereix mètode de matching, per la qual cosa qualsevol persona que ens busqui ens podrà enviar un missatge, ho desitgem o no. Els mateixos usuaris creen els esdeveniments i, si no hi ha cap report indicant que és fals, cap administrador els modera.

- ***Kickback***[3]: aplicació més orientada a organitzar esdeveniments amb els teus amics de *Facebook* que a conèixer-ne de nous. L'aplicació no compta amb servei de xat, però si amb servei de notificacions. No té mètode de matching. Té un bon sistema de geolocalització d'esdeveniments.

- ***Expreem***[4]: xarxa social semblant a *Facebook*. Ens permet fer sol·licituds d'amistat, donar likes a publicacions, unir-se a esdeveniments..., però no ofereix el sistema de matching selectiu.

Productes relacionats

- ***Tinder***[5]: potser l'aplicació més similar de totes pel que fa a funcionament i ja amb una quota de mercat molt establerta. *Tinder* és la mare de totes les aplicacions matchmaking i té un sistema de match similar al que estem dissenyant, el valor diferencial que oferirem és el matching selectiu: així com a *Tinder* el sistema de recomanació només es basa en l'edat dels usuaris i la distància entre ells, a *PartyPal* incloem el concepte d'esdeveniments en comú, que serveix com a filtre.

- ***Facebook***[6]: la xarxa social permet promoure esdeveniments dins de pàgines i convidar a amics i coneguts. Encara que és impossible competir en quota de mercat, *Facebook* té aquesta utilitat com una més i no té el sistema de matching selectiu.

- ***Badoo***[7]: competència directa de *Tinder* i també amb una quota de mercat molt establerta, superant els 400 milions d'usuaris. El funcionament és gairebé igual a *Tinder* i tampoc té el matching selectiu per esdeveniments en comú.

Proveïdors d'entrades

- ***Ticketmaster***[8]: ven les entrades dels esdeveniments i dona l'opció de compartir la pàgina en les teves xarxes socials, però no ofereix ni opcions de xat amb altres usuaris ni opció de matchmaking.
- ***Eventbrite***[9]: igual que *Ticketmaster*, ofereix la venda d'entrades, però no està pensat com a plataforma social.

Per tant, encara que veiem que hi ha aplicacions amb molt de pes i amb una quota de mercat elevada, queda un buit que no ha estat cobert com és el del matchmaking selectiu.

Pel que fa a la reutilització d'algun servei, no hi ha cap sistema ja en funcionament obert al públic que permeti realitzar la funció de matching que es necessita, per tant es dissenyarà un algorisme fet a mida per a les necessitats del projecte. Per a la funció del cercador, es farà servir el propi

de *Ticketmaster*, donada la seva potència i gratuïtat. Pel sistema de xat també s'optarà pel desenvolupament d'un propi per a poder integrar totes les funcionalitats desitjades, com és el sistema de notificacions personalitzades o els reports d'usuaris.

3. Abast del projecte

3.1. Formulació del problema

El problema que es vol atacar és la falta de serveis que ens ajudin a connectar amb gent nova a partir dels nostres gustos, que es representaran a partir dels esdeveniments que seguim. Com ja hem vist en el punt anterior, hi ha molt de mercat ja cobert en l'apartat de networking, però no hi ha cap que barregi la selectivitat i diversitat dels esdeveniments de *Facebook* i el mètode de socialització interactiva de *Tinder*. El que es busca en aquest projecte és imitar l'efecte "cua de concerts", on sense conèixer a les persones que tens davant i darrere, es dona peu a la conversa perquè que hi ha un tema en comú.

3.1.1. Objectius

Els objectius que es volen resoldre en el transcurs del projecte són els següents:

1. Cobrir un buit en el mercat d'aplicacions socials i matchmaking.
2. Desenvolupar un sistema de matching basat en bosses d'usuaris.
3. Implementar un sistema de xat en temps real.
4. Crear una aplicació atractiva, tan estèticament com funcionalment.
5. Usar tecnologies innovadores pel desenvolupament de la part pràctica del projecte.
6. Implementar un sistema d'integració contínua pel desplegament de l'aplicació.
7. Usar tècniques validades per a l'especificació, el disseny i la validació de tests i requisits.
8. Complir els objectius de qualitat esperats per a un Projecte de Final de Grau.
9. Finalitzar el projecte dins del termini establert i amb el mínim pressupost possible.

3.2. Abast

Tot i que les possibilitats tècniques i la imaginació ens podrien dur a planificar un sistema molt més extens i ambiciós del que es vol crear, la realitat és que com a treball de fi de grau estem sotmesos a unes limitacions inamovibles, com és la data de lliurament i el format individual de la tasca. És per aquest motiu que és necessari limitar l'abast que es cobrirà d'aquest projecte en aquest treball.

Els punts que es treballaran, a part de la memòria tècnica, seran els següents:

1. Desenvolupament d'un servidor back end en format *API-Rest* que administrarà el funcionament del sistema.
2. Desenvolupament d'una aplicació Android, que cobrirà la part de front end[4] i consumirà el servei de l'API. Es programarà l'aplicació per a Android, deixant per a un treball futur adaptar-la a altres sistemes operatius.
3. Integració del servei amb els sistemes de *Google* per a permetre l'autenticació dels usuaris, de *Ticketmaster* per a fer servir el seu sistema de cerques d'esdeveniments i de *Firebase realtime database* per a crear un sistema de xat i notificacions.

[4] Front end: capa de presentació d'un sistema, capa que interactua amb l'usuari, en terminologia informàtica. Gràficament, la carcassa exterior d'un rellotge de paret.

4. Implementació d'un sistema d'integració contínua per a agilitzar i facilitar el tràmit de desplegar el sistema en producció.

3.2.1. Obstacles

Els principals obstacles que limiten l'abast del projecte són els següents:

1. **Temps:** a més que el temps és limitat i l'elaboració del TFG s'ha de combinar amb altres aspectes de la vida quotidiana, és important remarcar que molts dels coneixements necessaris per a dur a terme aquesta pràctica són escassos o desconeguts inicialment pel programador, per tant, al temps de desenvolupament se li ha d'afegir el temps de formació necessari per a poder treballar amb aquestes eines amb naturalitat.
2. **Mà d'obra:** en ser un treball que té com a objectiu comprovar la correcta assimilació de tots els conceptes que s'han ensenyat durant el grau en Enginyeria Informàtica i es fa en format individual, el desenvolupador ha d'assolir tots els papers que en un equip més gran es repartirien entre diferents membres, per tant és de vital importància una perfecta organització i repartiment de les hores dedicables entre els diferents apartats que es volen tractar.
3. **Plataformes:** l'auge del telèfon mòbil i de diferents companyies competidores ha fet aparèixer una amalgama de sistemes operatius poc compatibles entre si. És per això que a l'hora de triar un sistema operatiu sobre el qual treballar ho hem de fer tenint present que només arribarem a una part de la població de smart phones.
4. **Acords amb empreses externes:** molts dels serveis que es faran servir tenen una versió gratuïta i limitada pel públic general i una amb més funcionalitats destinada a empreses partner i col·laboradors. Atès que el projecte es fa en el marc educatiu, acceptarem les limitacions amb les quals venen lligades les versions gratuïtes.
5. **Pressupost pel patrocini:** encara que no limita l'abast del projecte en si, si en acabar-lo es volgués introduir-lo al mercat, s'hauria de buscar pressupost per a poder generar publicitat i atreure usuaris.

4. Metodologia i gestió del projecte

4.1. Metodologia i rigor

4.1.1. Mètodes de treball

Encara que les metodologies agile estan pensades per a treballs en grup, són les que millor adapten la forma de treballar al tipus de projecte en el qual s'apliquen. S'utilitzarà una metodologia inspirada en *Scrum*. Les principals característiques de *Scrum* són l'adopció d'una estratègia de desenvolupament incremental a diferència d'una planificació i execució completa del producte, i el encavalcament de les diferents fases del desenvolupament, allunyant-se de metodologies seqüencials com podria ser el mètode en cascada. El desenvolupament del projecte es partirà en 3 fases:

La primera fase és la fase inicial, que es cobreix en l'assignatura de GEP (Gestió de Projectes). En aquesta es definirà el context del projecte, el seu estat de l'art, el problema a resoldre, l'abast de la solució, la planificació temporal i la gestió de la viabilitat econòmica i de sostenibilitat.

En la segona fase, es parteix el temps restant fins al lliurament final del projecte en *Sprints*. Els *Sprints* són períodes de temps fixos, per exemple de 2 setmanes, que es marquen al començament del projecte i no es poden canviar. Al començament de cada *Sprint* es defineixen quines de les tasques del product backlog (tots els casos d'ús que tenim a la llista To Do del nostre *Trello*) són més prioritàries per fer-les en la següent iteració. Cada cas d'ús té assignat un valor que indica el grau de complexitat que pot necessitar la tasca i quan es fa la tria de tots els casos que compondran un *Sprint* se sumen aquests valors. L'objectiu d'aquesta metodologia és intentar triar el nombre de tasques just per a no quedar-se curt de temps ni que en sobri, per tant, iteració rere iteració s'anirà refinant el valor aproximat de tasques/*Sprint*.

L'últim *Sprint* o mig *Sprint* abans de l'entrega serveix per a tancar el projecte i que tot estigui llest i funcional pel lliurament al client, en aquest cas, el tribunal de defensa del treball de final de grau.

4.1.2. Eines de seguiment

La primera eina de seguiment que es fa servir és *Trello* [10]. *Trello* fa servir el sistema kanban. A partir de la creació i la disposició de targetes virtuals entre diferents columnes, l'usuari pot organitzar tasques, agregar llistes, adjuntar arxius, etiquetar esdeveniments, fixar dates d'entrega, repartir les tasques entre els membres del taulell, etc. En aquest projecte, *Trello* es fa servir de la següent forma: el taulell el componen 4 llistes: To Do (Per fer), Doing (en procés), Done (acabat), If I have time (si tinc temps). En un començament es defineixen les tasques en l'àmbit general i se situen sota To Do o If I have Time, depenent si entren o no a l'abast especificat a la memòria. A mesura que es va avançant en el projecte, les targetes es mouen a Doing o Done, ajudant així a seguir el progrés general del projecte.

La segona eina que es fa servir és *GitHub*[11], plataforma que utilitza el sistema de control de versions GIT. *GitHub* s'usa com a repositori del codi i per a facilitar el treball entre màquines diferents. Si més endavant es veiés que cal establir una organització dins el repositori, es podria fer servir la metodologia *GitFlow*, que a partir de la creació i fusió de branques en format d'arbre, permet distingir entre el codi en producció i el codi encara en desenvolupament.

La tercera eina és *CircleCI*[12], gestor d'automatització del pipeline, o en altres paraules, el seguit de transformacions o processos pels quals va passant el codi d'ençà que surt de l'ordinador del programador fins que arriba a producció. En el cas d'aquest projecte, *CircleCI* està connectat al compte de *GitHub*, i per cada commit nou que es fa, agafa el codi, el construeix i li fa passar els tests especificats. Si res falla, es desplega a producció. En cas de fallada en algun dels passos, envia un missatge d'error al programador i avorta el procés de desplegament.

Per acabar, l'última eina de seguiment és la pròpia API, que en ser un servei en temps real permet provar al moment si s'han aplicat els últims canvis o si alguna cosa ha fallat.

4.1.3. Mètode de validació

Pel que fa a la memòria, les reunions regulars i l'intercanvi de correspondència electrònica freqüents amb la directora del projecte permetran validar que els artefactes que s'estiguin fent servir s'estiguin emprant de forma correcta.

Quant a la part pràctica, a més dels errors a simple vista que un sistema en temps real no pot amagar, es farà servir una metodologia de codi iterativa, on cada una de les funcions serà provada per diversos tests a la vegada que s'estigui programant. A part dels tests locals, el sistema d'integració contínua tindrà programats tests d'integració, que provaran que totes les peces del sistema funcionen correctament juntes, i s'executaran cada cop que es pugin fitxers al repositori de *GitHub*. Si algun d'aquests tests falla, el codi no passarà a producció i ens arribarà un correu electrònic indicant l'error i el seu motiu.

4.2. Planificació inicial

4.2.1. Consideracions globals

Abans de començar amb la descripció de cada una de les fases del projecte, és important remarcar que aquest s'està realitzant amb una metodologia inspirada en *Scrum*, on les diferents fases del treball es podran encavalcar i s'adoptarà una estratègia de desenvolupament incremental. És a dir, una mateixa tasca es podrà anar completant a mesura que avancin les iteracions, allunyant-se de la metodologia en cascada on no es pot avançar fins que la tasca anterior està finalitzada.

En metodologies *Scrum*, la divisió de feina es fa a partir de la identificació dels casos d'ús, que són la unitat mínima amb la qual es pot treballar. Aquests casos d'ús haurien de ser el més independent possible els uns dels altres per a ajudar a repartir la feina entre equips. Com aquest treball és individual, no és possible aplicar aquesta paral·lelització i la feina es realitzarà de forma seqüencial. És per això que s'ha optat pel diagrama de *Gantt* [13] (Annex 1), considerant innecessari el diagrama de PERT.

El desenvolupament del codi va lligat al dels tests, per tant, encara que no s'especifiqui textualment en cada cas d'ús, aquesta ve acompanyada dels seus corresponents tests, que es realitzaran en paral·lel.

Un altre aspecte que es realitzarà en paral·lel és el curs de GEP, que es realitzarà a la vegada amb el desenvolupament del treball en si.

La data d'inici del treball és el 20 de febrer de 2019 i la de finalització estimada és el 27 de juny de 2019. Aquesta data de finalització és orientativa a l'alça i inclou l'elaboració de la memòria, del curs de GEP, de la part pràctica i de la preparació de la defensa oral. L'estimació inicial en

hores és de 502 (91 dies), però donada la inexperiència amb moltes de les tecnologies usades, es podrien necessitar algunes més. Aquesta planificació s'ha realitzat pensant en jornades de treball de 6 hores diàries, 5 dies a la setmana, el que surt a 30 hores setmanals. La data de defensa és el 5 de juliol.

4.2.2. Planificació temporal

Com ja s'ha explicat, el projecte abastarà de mitjans de febrer a finals de juny. Es procedeix ara a explicar cada una de les iteracions i fases.

GEP

Aquesta fase cobreix el treball realitzat per a l'assignatura de GEP del 20 de febrer del 2019, moment que es comença el projecte, fins al 29 de març del mateix any, quan acaba el mòdul de l'assignatura de Gestió de Projectes. Aquesta etapa inclou l'abast del projecte i la contextualització, la planificació temporal, la gestió econòmica i de sostenibilitat, la presentació oral i l'elaboració del document final del mòdul.

Fase inicial

La fase inicial abasta del 12 de març del 2019 al 29 de març del mateix any. Inclou l'estudi i anàlisi dels requisits del sistema, la seva especificació i el disseny de tota l'arquitectura. Es realitzen totes aquestes tasques en aquesta fase perquè permeten assentar la base del projecte i ajudar a agilitzar el desenvolupament posterior.

Durant la fase inicial també es defineix el backlog [14] inicial. Aquest és un artefacte de les metodologies àgils que permet identificar totes les tasques a realitzar i ordenar-les per importància. A cada tasca se li assigna un pes i aquest pes es multiplica per dos per a saber el nombre d'hores aproximades a dedicar. En realitzar el backlog d'aquest projecte han sortit 251 punts de cas (pc), que s'han repartit entre 5 iteracions i una final. Cada iteració té assignats al voltant de 40pc i dues setmanes de temps. Si en aquest temps alguna de les tasques no es pot realitzar, el cas d'ús passarà a la següent iteració.

Iteracions

Cada iteració comença amb una petita planificació on s'estudia si la velocitat de feina és l'esperada i si s'han de realitzar modificacions sobre el full de ruta inicial. Procedeix amb el desenvolupament, i conseqüent testing, de diversos casos d'ús i acaba amb una demo a la tutora i l'actualització de la documentació necessària.

Primera iteració (01/04/2019 - 12/04/2019)

La primera iteració està preparada perquè el desenvolupador es familiaritzi amb tots els llenguatges, entorns de treball i eines que es faran servir durant el desenvolupament del projecte en si. També es programarà tot el front end necessari per a facilitar el treball de forma visual en següents iteracions.

Segona iteració (15/04/19 - 26/04/19)

Com en la primera iteració ja s'ha après tot el necessari per a poder desenvolupar el projecte, en la segona es pot començar a realitzar ja casos d'ús. S'espera poder finalitzar el sistema de registre i login, la gestió de les sessions dels usuaris, el sistema de perfils dels usuaris, el sistema

de configuració d'aquests perfils, on podran canviar la seva descripció i fotografies, el sistema de configuració del compte, on podran canviar la contrasenya, clau d'entrada… i el sistema d'esdeveniments, que començarà a preparar tota l'estructura sobre la qual es basarà la tercera iteració.

Tercera iteració (29/04/19 - 10/05/19)

Gran pes de la tercera iteració es troba en la gestió de l'obtenció d'esdeveniments a partir d'APIs externes com podria ser la de *Ticketmaster*. També es desenvolupa el cercador d'esdeveniments i la gestió de les subscripcions d'usuaris a aquests.

Quarta iteració (13/05/19 - 24/05/19)

La quarta iteració està plenament dedicada a l'elaboració de l'algoritme de recomanació, que determinarà les recomanacions d'altres usuaris que rebrà un usuari en concret. També es comença a desenvolupar el sistema de gestió de matchs, que permetrà a l'usuari desfer els que no li interessi mantenir i reportar els usuaris que no estiguin fent un ús correcte de l'aplicació.

Cinquena iteració (27/05/19 - 07/06/19)

En aquesta iteració es finalitza la tasca de gestió dels matchs i es treballa en el sistema de xat dels usuaris.

Fase final

En la fase final, que va del 10 de juny fins a l'acabament del treball, es migra tot el sistema al host de producció, deixant ja tot preparat i funcional per a la defensa final. També es redacten els apartats restants: validació dels requisits i conclusions. Aquesta fase també engloba la preparació de la defensa.

4.2.3. Diagrama de *Gantt*

Tal com indica la pàgina oficial Gantt.com [15], "Un diagrama de *Gantt*, utilitzat habitualment en la gestió de projectes, és una de les formes més populars i útils de mostrar activitats (tasques o esdeveniments) en funció del temps. A l'esquerra del gràfic hi ha una llista de les activitats i, al llarg de la part superior, hi ha una escala de temps adequada. Cada activitat està representada per una barra; la posició i la llargada de la barra reflecteixen la data d'inici, la durada i la data de finalització de l'activitat."

En el cas d'aquest projecte, en ser un treball individual totes les tasques es realitzen seqüencialment a excepció del treball de l'assignatura de GEP i la fase inicial, que es realitzen en paral·lel. La taula 1 resumeix el que es pot veure al diagrama de l'Annex 1. En aquest diagrama s'especifica el nom de la tasca, el seu pes en hores, la data inici i la seva data final, juntament amb una representació gràfica.

Tasca	Estimació del temps (en hores)	Dependències
1) GEP	61.1 h	-
2) Fase inicial	60 h	-
3) Primera iteració	60 h	2)
4) Segona iteració	60 h	3)
5) Tercera iteració	60 h	4)
6) Quarta iteració	60 h	5)
7) Cinquena iteració	60 h	6)
8) Iteració final	61 h	7)
9) Preparació defensa	20 h	8)
Total	502.1 h	

Taula 1: resum i interpretació del diagrama de Gantt de l'Annex 1

4.2.4. Valoració d'alternatives i pla d'acció

Durant el transcurs del projecte poden ocórrer diversos tipus de desviacions i és important estar preparat per a poder corregir-les a temps i que no afectin la qualitat de l'entrega final.

Desviació causada per una mala planificació temporal

Ja que aquesta planificació s'està realitzant a priori i s'està treballant en un escenari idíl·lic on s'aconsegueix fer cada una de les tasques en el temps establert, és molt probable que algun dels temps sigui inferior o superior al real. En cas que sigui inferior, no hi ha cap problema perquè donarà més marge a les altres tasques de *l'Sprint*. En cas que sigui superior, s'haurà de contrarestar treballant més hores.

El pla de contenció per a evitar que en un *Sprint* no s'acabi les tasques planificades és el de treballar també els dissabtes, afegint 12 hores de desenvolupament més a la iteració (cada iteració té 2 dissabtes, i es treballaria 6 hores al dia).

Desviació causada per dificultats tècniques

Ja que moltes de les tecnologies que es faran servir no són conegudes pel programador, s'ha donat un marge gran per a l'aprenentatge i familiarització d'aquestes. Tot i així, durant la fase inicial es podria començar a estudiar el funcionament de les eines per a descarregar de feina la primera iteració. També es podrien fer servir els caps de setmana si ens adonem que el temps dedicat a l'aprenentatge ha estat insuficient.

Desviació causada per imprevistos

A banda de les dues desviacions esmentades, no es pot tenir un control absolut de l'entorn en el qual vivim/treballem. És important ser conscient que encara que es tingui la planificació més exacta, és vital ser flexible i estar preparat per als imprevistos. És per aquest motiu que es deixen fora de la planificació els caps de setmana de la planificació, però es podria comptar amb ells si fos necessari.

4.3. Recursos

Cada projecte compta amb un seguit de recursos que pot fer servir per tal d'acabar el treball en el temps i qualitats acordats amb el client. En el cas d'aquest TFG, es conta amb un seguit de recursos personals, materials i de software que es defineixen a continuació.

Recursos personals

De cara als recursos personals, l'estudiant dedicarà 30 hores setmanals inicialment. Si durant el transcurs del projecte es veu que no s'està seguint la planificació temporal i les tasques s'estan endarrerint, es pujarà a una dedicació de 36 hores setmanals, fent servir els caps de setmana. A més, s'afegeix l'ajut de les correccions i el guiatge de la tutora, la Cristina Gómez, i totes les possibles consultes a companys que responen de forma desinteressada.

Recursos materials

Per a la realització d'aquest projecte es requereix un seguit de recursos materials:

Serà necessari un lloc de treball físic per a dur a terme el projecte. Aquest lloc de treball pot ser la Universitat o la pròpia residència de l'estudiant, depenent de la seva disponibilitat en aquell moment. Caldrà tenir en compte costos com l'electricitat.

Per a la redacció de la memòria i la implementació del sistema caldrà disposar d'un ordinador, sigui portàtil o de sobretaula. Aquest dispositiu haurà de tenir instal·lat tot el software necessari per a poder complir les expectatives del treball, així que serà necessari incloure, si existís, el cost de tot el programari que es farà servir.

A més a més, com el treball a realitzar és una aplicació al cloud, cal afegir el cost de servidors i sistemes que es faran servir per a emmagatzemar de manera remota la nostra aplicació. Es faran servir *Digital Ocean* i *Firebase* principalment.

Recursos de software

Per a l'elaboració d'aquest treball serà necessari utilitzar divers programari software.

El primer gran bloc és el conjunt d'aplicacions *Google Apps* [16]. Aquest grup inclou *Drive, Docs, Sheets, Slides i Draw*s. Totes aquestes funcionalitats de *Google* es faran servir per a desenvolupar la memòria i exposicions orals, i emmagatzemar tot el material en un mateix lloc. Com a complement a *Google Drive*, també s'utilitzarà *Microsoft Office Word*.

Per a la confecció de diagrames com el de *Gantt*, s'emprarà l'eina *SmartSheet* [17].

Per al desenvolupament de l'aplicació, s'usarà *Android Studio* per a la confecció de l'app mòbil i *Intellij Idea* pel back end, ambdós productes de *Jet Brains* [18]. A més a més, es farà servir *Docker* per als contenidors virtuals, *GitHub* per al sistema de versions en *GIT* i *CircleCI* per al desplegament continu en les màquines de *Digital Ocean* [19].

Per últim, per a aplicar una metodologia agile d'una manera més directa farem servir *Trello*, que a partir d'un sistema de targetes ens ajudarà a organitzar cada una de les iteracions.

4.4. Gestió econòmica

Tot i que el treball final de grau és un projecte no remunerat, sempre és important confeccionar un pressupost del que costaria realment. Això ajudarà a determinar si el projecte és viable o no, o si cal retallar-ne l'abast. L'equip d'aquest treball el forma una única persona que assumeix tots els rols, però en una situació real no seria així.

És per això que per a la confecció d'aquesta gestió econòmica se suposa l'existència de 5 rols: el cap de projecte, l'analista, el dissenyador, el programador i el tester, que s'encarregaran de realitzar totes les proves al sistema.

4.4.1. Identificació i estimació de costos

Per a la construcció del pressupost, cal tenir en compte diversos apartats necessaris per a poder desenvolupar un projecte[20]:

El primer són els costos directes, que són aquells directament imputables a l'execució d'una activitat. Exemples podrien ser els recursos humans, materials, etc. A aquests els segueixen els costos indirectes, que no són directament atribuïbles a l'activitat que estem realitzant, però igualment necessaris. Per exemple, el preu de la llum, del local de treball, de la connexió a Internet, de l'amortització dels equips, etc. A aquests dos costos se li suma el d'imprevistos, que permet atenuar errors de l'estimació o descuits. Normalment es calcula com el 10-20% de la suma de costs directes i indirectes. Per últim la partida de contingència, on a partir de la probabilitat estimada que una acció no planejada ocorri i el cost que tindria, s'assigna un valor per a corregir-la.

Si la partida d'imprevistos no s'arribés a necessitar, el seu valor s'afegiria a la de contingències. Si al final aquesta última tampoc calgués, es destinaria el pressupost sobrant a millorar la infraestructura interna, com podrien ser servidors o llicències, per a preparar el terreny de treball futur.

La suma de tots aquests punts esdevindrà el pressupost final del projecte.

4.4.1.1. Costos directes

La consultora Michael Page [21] realitza un estudi anual de les tendències del mercat laboral a Espanya en l'àmbit tecnològic [22]. L'estudi indica la mitjana del sou mínim i la del sou màxim en els diversos rols que conformen un projecte. Basant-se en aquestes dades, agafarem la mitjana de les dues com a valor d'estudi. Els sous estan indicats en valor anual. Per a obtenir la remuneració per hora els dividim per 1760, donat que de mitjana un any té 220 dies laborables i jornades de 8 hores diàries.

Els sous dels diferents rols del projecte es mostren en la taula 2:

Rol	Sou mínim mitjà	Sou màxim mitjà	Sou mitjà	Remuneració per hora
Cap de projecte	40.000€	60.000€	50.000€	28,41€
Analista	24.000€	45.000€	34.500€	19,60€
Dissenyador	18.000€	30.000€	24.000€	13,64€
Programador i tester	18.000€	40.000€	29.000€	16,48€

Taula 2: sou mitjà espanyol dels diferents rols implicats en el projecte

Un cop definits els sous que s'assignaran a cadascun dels rols que intervenen en la confecció del projecte, s'estudien les tasques definides anteriorment en el diagrama de *Gantt* en la taula 3 per a trobar el cost directe deduïble dels recursos humans:

Fase (dedicació estimada)	Rol (dedicació en hores)				Total
	Cap de projecte	Analista	Dissenyador	Programador i tester	
GEP (61.1 h)	61,1				1.735,80€
Fase inicial (60 h)		30	30		997,16€
Primera iteració (60 h)	5		4	51	1.036,93€
Segona iteració (60 h)	2		2	56	1.006,82€
Tercera iteració (60 h)	2		2	56	1.006,82€
Quarta iteració (60 h)	2		2	56	1.006,82€
Cinquena iteració (60 h)	2		2	56	1.006,82€
Iteració final (61 h)	22		8	31	1.244,89€
Preparació defensa (20 h)	20				568,18€
Total	116,1	30	50	306	**9.610,23€**

Taula 3: cost per rol de cadascuna de les fases del projecte

4.4.1.2. Costos indirectes

Per al còmput dels costos indirectes es farà servir el cost unitari. D'aquesta manera és més fàcil determinar quin d'aquest cost és deduïble a la realització del treball.

Com que es necessita un lloc per a treballar, s'han d'incloure el valor de les factures del lloguer, Internet, llum… corresponent a les hores de dedicació. També s'ha d'incloure el valor de l'abonament del transport [23]. Com aquest va ser adquirit amb la finalitat de transportar l'equip al lloc de treball, aniria a càrrec dels costos indirectes.

En aquesta partida també s'ha d'incloure el valor de les amortitzacions dels equips de treball. Hisenda determina que un equip informàtic té una vida útil de 3-4 anys, i que després d'aquest

temps ja està amortitzat [24]. Per últim, cal afegir el preu del servidor que es farà servir per a desplegar el projecte en producció.

La taula 4 reflecteix i detalla els 513,8€ resultants:

producte	Cost unitari	Unitats	Cost deduïble
Llum, Internet, local...	0.347€/hora	502 hores	174,30€
Transport	244€	1 abonament	244,00€
Amortització sobretaula	0€/hora	250 hores	0,00€
Amortització portàtil	0,075€/hora	252 hores	23,00€
Amortització Smartphone	0,075€/hora	502 hores	47,50€
Servidors *Digital Ocean*	5€/mes	5 mesos	25,00€
Total			**513,80€**

Taula 4: costos indirectes del projecte desglossats per activitat i cost unitari

4.4.1.3. Imprevistos

Ja s'ha explicat anteriorment que la reserva d'imprevistos actua com a fons de reserva o "matalàs" per a situacions que no s'ha tingut en compte mentre es confeccionava la planificació econòmica. Es calcula a partir de multiplicar la suma de costos directes i indirectes per un valor que oscil·li entre el 10 i el 20%. En la Taula 5 es pot comprovar que per a aquest projecte s'agafarà el valor mitjà, 15%.

Activitat	Cost	Percentatge	Cost total
Costos directes	9.610,23€	15%	1.441,53€
Costos indirectes	513,80€	15%	77,07€
Total			**1.518,60€**

Taula 5: reserva d'imprevistos a partir dels costos directes i indirectes

4.4.1.4. Contingències

Donada la falta d'experiència en la planificació de projectes, és probable que durant el transcurs d'aquest succeeixin situacions de risc que puguin afectar el pressupost.

La primera i la més probable és que no s'aconsegueixi acabar el projecte en les hores programades i sigui necessari realitzar-ne de més. S'ha valorat que hi ha un 50% de probabilitats d'haver de necessitar 50 hores més de desenvolupament per part del programador i tester. Per tant, el cost addicional seria de 350€.

La segona situació de risc seria una avaria en el hardware que obligaria a realitzar una reparació perquè els dispositius ja es troben fora de garantia. Per pal·liar aquest imprevist que s'estima que podria ocórrer en un 5% dels casos, s'afegeix un cost de contingència de 30€.

És important recordar que el cost de contingències es calcula a partir de la probabilitat d'ocurrència d'aquest multiplicat pel seu preu unitari. La taula 6 mostra el detall d'aquesta partida.

Imprevist	Probabilitat	Unitats	Preu	Cost
Endarreriment planificació inicial	50%	50 hores	14€/hora	350,00€
Avaria en el hardware	5%	1	600€	30,00€
Total				**380,00€**

Taula 6: reserva de contingències per a pal·liar situacions de risc

4.4.1.5. Pressupost final

Per tant, el pressupost final es compon de cadascuna de les partides explicades en els apartats anteriors. Si a més es volgués obtenir un benefici, s'hauria de calcular a partir de multiplicar els 12.022€ de la Taula 7 pel marge de guany desitjat.

Activitat	Import
Costos directes	9.610,23€
Costos indirectes	513,80€
Imprevistos	1.518,60€
Contingències	380,00€
Total	**12.022,63€**

Taula 7: pressupost final del projecte

4.4.2. Control de gestió

La mateixa metodologia ja funciona com a controlador de gestió: al començament de cada iteració es dedica una hora a la planificació. En aquesta es veurà com es va acabar l'anterior i si cal fer modificacions en la planificació inicial. En ser iteracions de dues setmanes, el marge de correcció és ràpid. A la vegada, cada *Sprint* té destinades 2 hores per a fer una o més reunions amb la tutora, que també servirà perquè supervisi que tot s'està fent correctament i en els terminis definits.

A part de la metodologia, es durà un historial d'hores dedicades que s'haurà d'actualitzar diàriament. El document inclou la data, el nom de la tasca que s'ha realitzat, una opció per a indicar si aquesta estava dins del bloc de documentació o implementació i les hores dedicades. Aquest document té certs indicadors automàtics que informen si el ritme que s'està seguint és l'adequat o cal incrementar el nombre d'hores.

Per últim, l'eina *Trello* permet especificar deadlines a les tasques. Un cop ha passat la data, apareix un indicador d'advertència al costat de la targeta que serveix per a ser conscients que estem fora del temps programat.

5. Anàlisi de requisits

L'estàndard IEEE std 610.12-1990 [25], que defineix termes de l'Enginyeria del Software, estableix la definició de Requisit com "Una condició o capacitat necessitada per un usuari per a resoldre un problema o assolir un objectiu" i com "Una condició o capacitat que ha de complir o posseir un sistema o un component d'aquest per tal de satisfer un contracte, un estàndard, una especificació o altres documents imposats formalment."

Realitzar una correcta anàlisi de requisits és una tasca vital per entendre la visió i el context del projecte: per assegurar-nos que una solució software resol correctament un problema, primer hem d'entendre i definir quin problema necessita ser resolt.

El propi estàndard IEEE std 610.12-1990 defineix anàlisi de requisits com "El procés de l'estudi de les necessitats dels usuaris per arribar a una definició dels requisits del sistema, de maquinari o de programari."

Encara que pugui semblar una tasca trivial, pot resultar realment difícil descobrir, entendre i formular quin problema hem de resoldre, el perquè aquest problema ha de ser resolt i el qui està implicat en resoldre'l. És important remarcar que en l'anàlisi de requisits es defineixen les funcions que el sistema ha de proporcionar, però no com ho farà. El com es detalla a l'apartat de disseny del sistema.

En aquest apartat analitzarem dos tipus de requisits diferents: els funcionals i els no funcionals.

Els requisits funcionals són les accions que el producte ha de ser capaç de proporcionar si vol ser d'utilitat per als clients. Aquest tipus de requisits emergeixen de les funcionalitats que els stakeholders necessiten realitzar.

Els requisits no funcionals, o requisits de qualitat, són propietats o qualitats que el producte ha de tenir. Especifiquen els criteris que es poden utilitzar per jutjar el funcionament d'un sistema, en lloc del comportament específic.

5.1. Requisits funcionals

Es detallen a continuació els requisits funcionals i una breu descripció del que el sistema hauria de proporcionar per satisfer-los.

Usuaris: Gestió de l'accés al compte

- **Log in/ Log out:** el sistema ha de permetre a l'usuari entrar o sortir del sistema amb les credencials especificades durant el registre del compte.
- **Log in amb *Google*:** el sistema ha de permetre iniciar sessió amb un compte de *Google*.
- **Registre:** el sistema ha de permetre a l'usuari registrar un compte per a fer servir l'aplicació. Li caldrà especificar nom, cognom, edat, gènere, email i contrasenya.
- **Registre amb *Google*:** el sistema ha de permetre crear un compte fent servir el registre per *Google*, on la recopilació de dades que se sol·licitaran a aquest servei es farà de forma transparent a l'usuari.

- **Recuperació de contrasenya:** el sistema ha d'estar preparat en el cas que l'usuari no recordi la contrasenya del seu compte i pugui demanar recuperar-la a partir d'un codi enviat al correu electrònic associat al compte.

Usuaris: Gestió d'esdeveniments

- **Buscar esdeveniment per nom al buscador:** el sistema ha de permetre a l'usuari cercar un esdeveniment en el qual pot estar interessat. En el cas que existeixi, el sistema li retorna la informació sobre aquest i la possibilitat de seguir-lo. Seguir un esdeveniment indica al sistema que l'algoritme de match podrà mostrar-li a l'usuari altres usuaris que també segueixin aquest esdeveniment.

- **Seguir un esdeveniment:** un cop l'usuari trobi un esdeveniment en el qual estigui interessat, el sistema ha de permetre-li seguir-lo. Aquesta acció es tradueix en el fet que tota la gent que també hagi seguit aquest esdeveniment, podrà ser recomanada a l'usuari quan aquest estigui fent servir l'algoritme de match.

- **Deixar de seguir un esdeveniment:** l'usuari perd l'interès en un esdeveniment o ja no li interessa que li apareguin usuaris relatius a aquest i el sistema ha de permetre-li deixar de seguir-lo.

- **Llistar els esdeveniments que segueixes:** l'usuari podrà accedir a la finestra d'esdeveniments, on el sistema li mostrarà tots els que ja segueix.

Usuaris: Gestió del mètode de recomanació

- **Mostrar usuaris recomanats:** si quan l'usuari u1 vol veure quines recomanacions d'altres usuaris li fa el sistema basant-se en els esdeveniments que segueix activa aquesta funcionalitat, el sistema li haurà d'anar recomanant perfils i l'usuari u1 els podrà acceptar o rebutjar. Depenent del que hagi respost u2 si u1 ha acceptat la recomanació, es generarà match o no.

- **Enviar un missatge a un usuari que ens interessa:** si quan l'usuari u1 està fent servir l'algoritme de recomanació, és a dir, si quan el sistema li va recomanant perfils d'usuaris d'esdeveniments que segueix, es troba amb algun perfil que li interessa, diguem el de l'usuari u2, el sistema ha de permetre-li enviar-li un missatge a u2. Si u2 respon, es crea match. Si rebutja el missatge, el match es donaria per invàlid i desapareixeria el missatge de la finestra de xats del receptor u2.

- **Rebutjar un usuari que no ens interessa:** si quan l'usuari u1 està fent servir l'algoritme de recomanació li apareix un perfil que no li interessa, el sistema ha de permetre rebutjar-lo, mostrant-li un perfil d'un altre usuari.

- **Guardar un usuari que ens pot interessar:** si quan l'usuari u1 està fent servir l'algoritme de recomanació li apareix un perfil que li podria interessar, diguem el de l'usuari u2, però no tant com per a enviar-li un missatge, el sistema ha de permetre-li guardar el seu perfil. Si l'usuari u2 guardés a u1 o li enviés un missatge perquè el sistema li ha recomanat el perfil i li ha agradat, es generaria match, si no, no.

- **Navegar pel perfil d'un usuari:** quan l'algoritme de recomanació en mostra una recomanació, l'usuari ha de poder navegar entre les seves fotografies i la seva descripció.

- **Respondre a un missatge de match:** si l'algoritme de recomanació li ha mostrat el nostre perfil a algú i aquest ha decidit enviar-nos un missatge i nosaltres estem interessats en aquesta persona, el sistema ha de permetre respondre el missatge generant match.

- **Rebutjar un missatge de match:** si quan a l'usuari u2 l'algoritme de recomanació li ha mostrat el nostre perfil (u1) i aquest ha decidit enviar-nos un missatge i nosaltres no estem interessats en aquesta persona, solicitarem cancel·lar el match i el sistema haurà de donar-lo per invàlid. Quan això succeeix, el xat desapareixerà de la llista de missatges del receptor u1.

- **Llistar xats:** el sistema ha de mostrar tots els xats amb matches que té un usuari i el contingut d'aquests.

- **Xatejar amb un usuari que ja és match:** un cop es té match amb un usuari, s'hagi aconseguit de qualsevol de les formes descrites anteriorment, el sistema haurà de permetre xatejar amb ell tant com es vulgui, donat que el xat no desapareixerà.

- **Desfer un match:** L'usuari no vol seguir tenint match amb qualsevol altre usuari i el sistema haurà de permetre desfer-lo. Aquesta acció evita que puguin seguir xatejant i desapareixeran de la llista de xats de l'altre usuari.

- **Reportar un altre usuari:** Si l'usuari u1 troba que la conducta d'un determinat usuari u2 és inadequada, ha d'existir l'opció de reportar-lo al sistema.

Usuaris: Gestió del compte

- **Actualitzar les dades del nostre perfil:** si un usuari troba que la seva descripció o les dades que mostra a altres usuaris són insuficients, o simplement vol afegir o suprimir alguna part d'aquestes, el sistema haurà de permetre modificar-les.

- **Actualitzar les fotografies del nostre perfil:** el sistema haurà de permetre a l'usuari afegir una fotografia al seu perfil si encara no ha arribat al límit permès o eliminar-ne alguna de les que ja es mostren.

- **Modificar la configuració del compte:** el sistema haurà de permetre a l'usuari canviar el seu correu, data de naixement o contrasenya.

- **Modificar les preferències de l'algoritme de recomanació:** el sistema haurà de permetre a l'usuari tenir control del gènere dels perfils què li recomanarà l'algoritme de recomanació.

5.2. Requisits no funcionals

Per a fer l'anàlisi de requisits no funcionals ens basarem en la classificació de Volere [26], que ajuda a organitzar els requisits atenent al seu domini.

Per a cada requisit no funcional determinat, s'especifica a quina classificació de Volere pertany, una descripció del requisit, justificació de la seva importància i el criteri de satisfacció que es farà servir per a provar si s'ha assolit o no.

Tipus de requisit de Volere	11b. Personalization and Internationalization
Número de requisit	1
Descripció	El sistema ha d'estar disponible en més d'una llengua.
Justificació del requisit	El sistema ha d'oferir les llengües més parlades, amb possibilitat d'ampliar-les amb el temps.
Criteri de satisfacció	S'oferirà l'app en català, castellà i anglès.

Tipus de requisit de Volere	11b. Personalization and Internationalization
Número de requisit	2
Descripció	El sistema ha de recordar la llengua triada per l'usuari.
Justificació del requisit	El sistema ha de guardar la tria de llengua de l'usuari.
Criteri de satisfacció	Es provarà que si es canvia la llengua i es tanca l'app, un cop es torni a obrir, la llengua en què es mostra és la que s'havia seleccionat. S'espera un encert del 100% dels casos.

Tipus de requisit de Volere	11c. Learning
Número de requisit	3
Descripció	Ha de ser fàcil aprendre com fer servir el sistema després de completar un petit tutorial.
Justificació del requisit	La primera vegada que l'usuari accedeixi a l'aplicació després d'haver-la instal·lat, se li oferirà un petit tutorial introductori que farà un recorregut per l'app explicant com funciona aquesta.

Criteri de satisfacció	S'espera que el 80% dels usuaris que realitzin el test el superin satisfactòriament després de completar un petit tutorial.

Tipus de requisit de Volere	11d. Understandability and Politeness
Número de requisit	4
Descripció	Les icones i les formes usades han de ser indicatius de la secció que representen.
Justificació del requisit	L'usuari ha de ser capaç d'associar les funcionalitats de l'app a les icones que les representen.
Criteri de satisfacció	S'espera que el 80% dels usuaris que realitzin el test sàpiguen orientar-se en l'aplicació a partir de les icones triades per a cada finestra.

Tipus de requisit de Volere	12c. Precision or Accuracy
Número de requisit	5
Descripció	L'algoritme de recomanació només pot mostrar usuaris d'esdeveniments que segueixes.
Justificació del requisit	Un usuari segueix un esdeveniment perquè el sistema li mostri gent que també està interessada. És important que el sistema realitzi aquesta acció correctament donat que és un dels atractius més importants del producte.
Criteri de satisfacció	L'algoritme de recomanació només mostrarà usuaris d'esdeveniments que l'usuari segueix. S'espera que el 100% dels usuaris que realitzin el test només es trobin amb usuaris d'esdeveniments que segueixen.

Tipus de requisit de Volere	13c. Requirements for Interfacing with Adjacent Systems
Número de requisit	6
Descripció	El sistema ha de funcionar en la majoria de dispositius Android.
Justificació del requisit	És important que l'aplicació sigui usable tant en telèfons d'última generació, com els no tan nous. A la vegada, també és important triar una versió

	d'Android que suporti les últimes innovacions de la plataforma.
Criteri de satisfacció	El sistema farà servir una versió d'Android disponible en més del 70% de dispositius Android. El sistema farà servir una versió d'SDK 23, corresponent a la versió d'Android 6.0 Marshmallow, que funciona al 71% dels dispositius [27]

Tipus de requisit de Volere	13d. Productization
Número de requisit	7
Descripció	El sistema ha de ser distribuït des de la Play Store.
Justificació del requisit	Les plataformes oficials són la manera més fàcil i fiable perquè els usuaris descarreguin contingut.
Criteri de satisfacció	El sistema ha de passar els controls de la plataforma *Google Play* i ha d'estar disponible per a la seva descarrega sempre en l'última versió alliberada.

Tipus de requisit de Volere	15a. Access
Número de requisit	8
Descripció	Les contrasenyes de la base de dades seran encriptades.
Justificació del requisit	Sempre és bona idea aportar un nivell extra de seguretat al sistema. Aquest es trobaria en el nivell de la base de dades.
Criteri de satisfacció	Totes les contrasenyes emmagatzemades a les bases de dades del sistema hauran de ser encriptades per tal d'augmentar la seguretat.

6. Especificació

Un cop s'ha realitzat l'anàlisi de requisits i hem entès quines són les exigències del client, es procedeix a descriure el comportament extern del sistema des del punt de vista de l'usuari i de l'entorn. L'objectiu d'aquest capítol és, a partir de l'ús de determinats artefactes, especificar d'una manera tècnica els requisits, agrupar-los depenent de la seva funció i ajudar a delimitar la frontera del projecte.

6.1. Diagrama de casos d'ús

El primer artefacte que es farà servir és el diagrama de casos d'ús. Un cas d'ús és un "document que descriu una seqüència d'esdeveniments que realitza un actor (agent extern) que usa el sistema per dur a terme un procés que té algun valor per a ell" [28].

El diagrama de casos d'ús representa gràficament els casos d'ús del sistema, els actors que intervenen i les relacions entre els actors i els casos d'ús.

Usuaris: Gestió de l'accés al compte

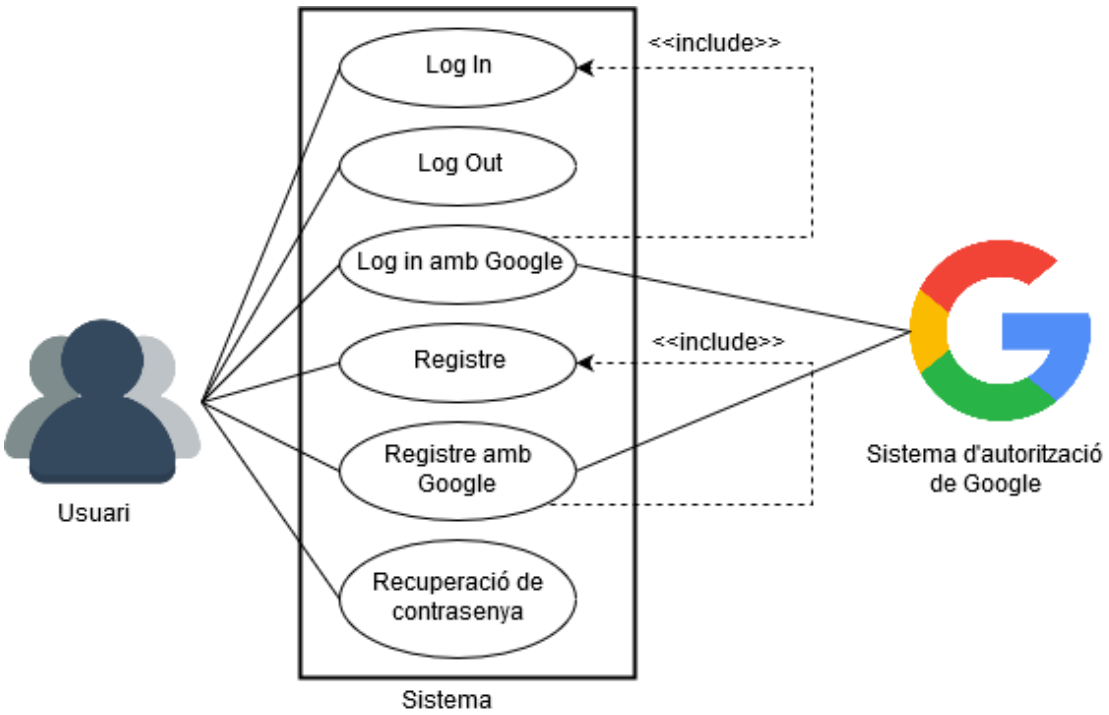


Figura 1: diagrama de casos d'ús per a la gestió de l'accés al compte dels usuaris.

Usuaris: Gestió d'esdeveniments

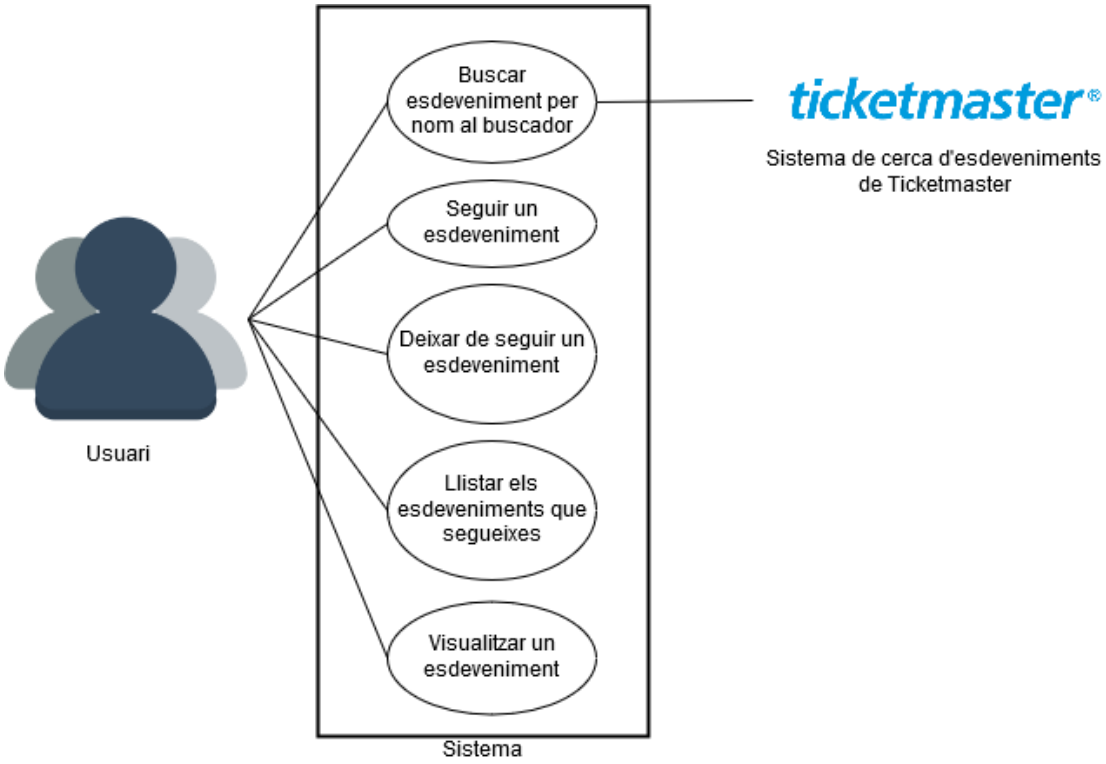


Figura 2: diagrama de casos d'ús per a la gestió d'esdeveniments dels usuaris.

Usuaris: Gestió del compte

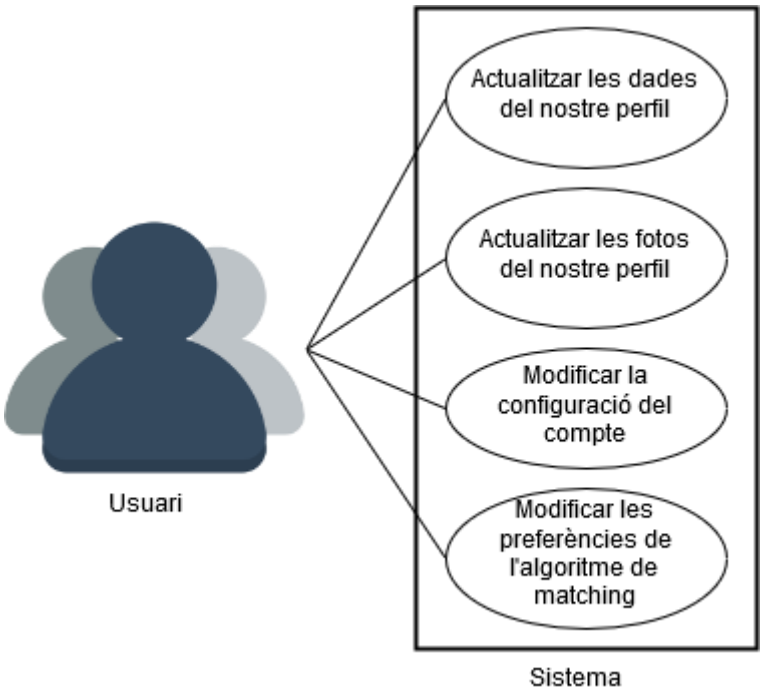


Figura 3: diagrama de casos d'ús per a la gestió del compte dels usuaris.

Usuaris: Gestió del mètode de recomanació

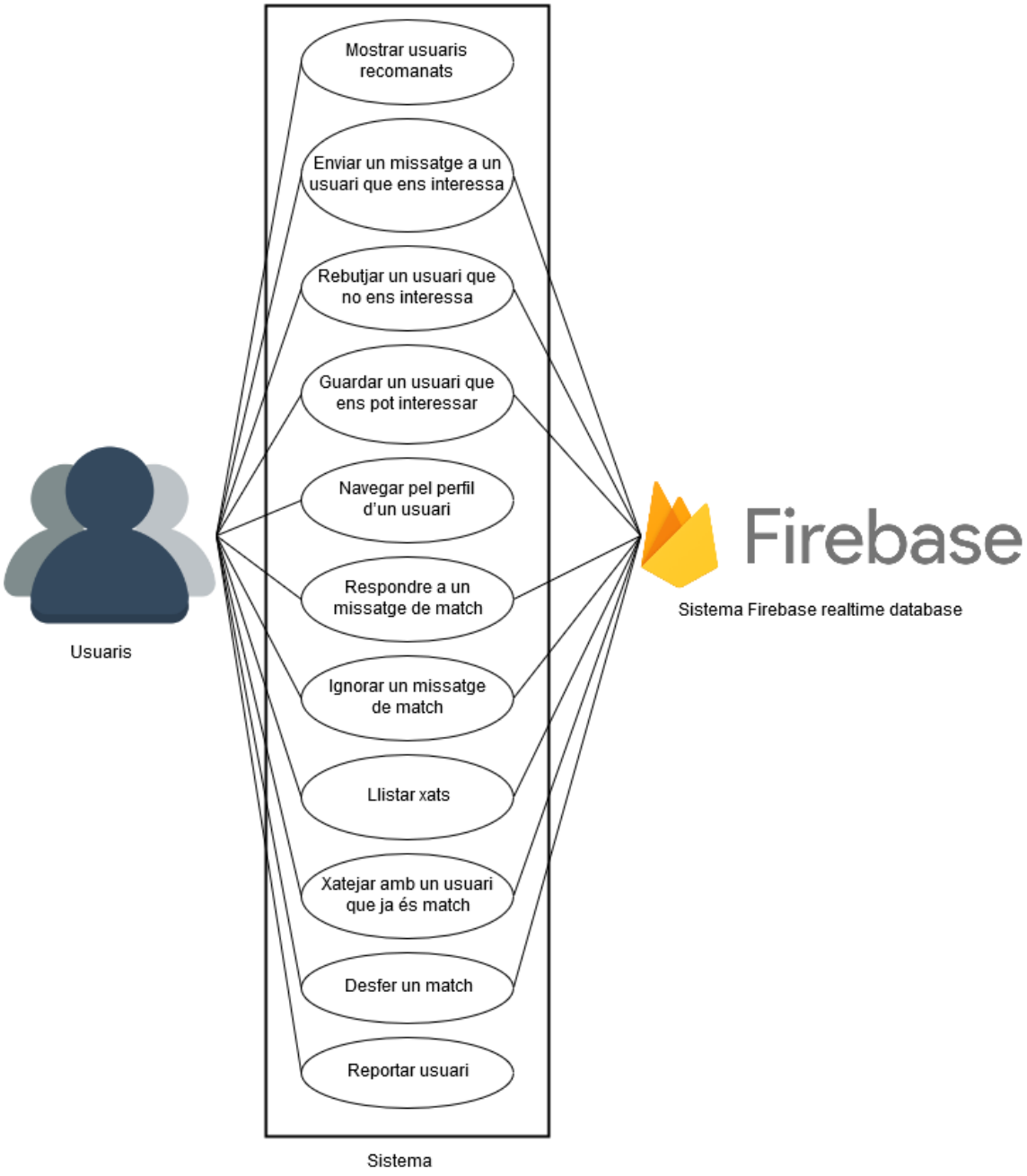


Figura 4: diagrama de casos d'ús per a la gestió del mètode de recomanació dels usuaris.

6.2. Descripció de casos d'ús

Aquest apartat descriurà en profunditat cada cas d'ús a implementar pel sistema. Cada cas d'ús vindrà definit en una taula que seguirà la següent estructura: nom del cas d'ús, actor del cas d'ús, precondicions del cas d'ús (totes aquelles condicions implícites que s'han de complir quan el cas d'ús s'executi), disparador (acció que activa el cas d'ús), escenari principal d'ús o escenari d'èxit (seguit d'accions que se succeeixen en el cas que tot funcioni correctament) i escenaris alternatius o extensions (seguit d'accions que se succeeixin en el cas que alguna acció de l'escenari principal no hagi funcionat correctament).

Usuaris: Gestió de l'accés al compte

Cas d'ús	**#1 Log in**	**Actor principal**	Usuari
Precondicions	-		
Disparador	L'usuari vol accedir al seu compte del sistema		
Escenari principal d'ús			
1. L'usuari introdueix el correu i contrasenya i indica al sistema que validi les credencials. 2. El sistema valida les credencials i retorna un codi d'acceptació. 3. El sistema redirigeix l'usuari a la primera pantalla de dins l'app.			
Extensió: contrasenya incorrecta			
4. El sistema determina que les credencials no són correctes. 5. El sistema avisa l'usuari que les credencials no són correctes a partir d'un codi, permetent a l'usuari tornar a intentar-ho. 6. El sistema recomana a l'usuari recuperar la contrasenya o crear un compte.			

Cas d'ús	**#2 Log in amb *Google***	**Actor principal**	Usuari
Precondicions	-		
Disparador	L'usuari vol accedir al seu compte del sistema fent servir el seu compte de *Google*		
Escenari principal d'ús			
1. L'usuari indica al sistema que vol fer log in amb el seu compte de *Google*. 2. El sistema es comunica amb el servei de *Google*. 3. El servei de *Google* li retorna les credencials que necessita. 4. El sistema retorna un codi d'acceptació a l'usuari. 5. El sistema redirigeix l'usuari a la primera pantalla de dins l'app.			

Extensió: credencials errònies
3. El servei de *Google* retorna un error. 4. El sistema avisa l'usuari que les credencials no són correctes, permetent a l'usuari tornar a intentar-ho. 5. El sistema recomana a l'usuari recuperar la contrasenya o crear un compte.

Cas d'ús	**#3 Log out**	**Actor principal**	Usuari
Precondicions	L'usuari està loguejat		
Disparador	L'usuari vol tancar la sessió del seu compte del sistema		
Escenari principal d'ús			
1. L'usuari indica al sistema que vol tancar sessió. 2. El sistema el retorna a la pantalla de login.			

Cas d'ús	**#4 Registre**	**Actor principal**	Usuari
Precondicions	-		
Disparador	L'usuari vol crear un compte en el sistema		
Escenari principal d'ús			
1. L'usuari omple el formulari de registre i indica al sistema que creï un compte. 2. El sistema valida les dades introduïdes per l'usuari. 3. El sistema guarda les dades introduïdes per l'usuari. 4. El sistema indica l'usuari que el compte ha estat creat. 5. El sistema redirigeix l'usuari a la vista de login.			
Extensió: clau primària en us			
4. El sistema valida les dades introduïdes per l'usuari i veu que la clau primària ja es troba en ús. 5. El sistema avisa l'usuari que aquesta clau primària ja es troba en ús i recomana recuperar la contrasenya. a. L'usuari introdueix una altra clau primària. b. L'usuari demana recuperar la contrasenya del compte que vol fer servir. Comença cas d'ús "Recuperació de contrasenya".			

Cas d'ús	**#5 Registre amb *Google***	**Actor principal**	Usuari
Precondicions	-		
Disparador	L'usuari vol crear un compte en el sistema fent servir el seu compte de *Google*		

Escenari principal d'ús

1. L'usuari indica al sistema que vol registrar-se amb el seu compte de *Google*.
2. El sistema es comunica amb el servei de *Google*.
3. El servei de *Google* li retorna les credencials que necessita.
4. El sistema guarda les dades rebudes del servei de *Google*.
5. El sistema indica l'usuari que el compte ha estat creat.
6. El sistema redirigeix l'usuari a la vista de login.

Extensió: clau primària en us

4. El sistema avisa l'usuari que aquesta clau primària ja es troba en ús i recomana recuperar la contrasenya.
    a. L'usuari selecciona un altre compte de *Google* per fer el registre i es torna al pas 2.

Cas d'ús	**#6 Recuperació de contrasenya**	**Actor principal**	Sistema
Precondicions	-		
Disparador	L'usuari vol recuperar la contrasenya d'un compte de la seva autoria		

Escenari principal d'ús

1. L'usuari vol recuperar la seva contrasenya i li indica al sistema la clau primària que es feia servir per a iniciar sessió en el compte.
2. El sistema valida que la clau primària existeix.
3. El sistema envia un mail de recuperació al mail associat a aquest compte.
4. L'usuari copia el codi que s'inclou al mail i l'introdueix al sistema, juntament amb l'antiga clau primària i la nova contrasenya i li indica al sistema que la canviï.
5. El sistema valida que el codi sigui vàlid.
6. El sistema guarda la nova contrasenya i redirigeix l'usuari a la vista de login.

Extensió: clau primària no existeix

3. El sistema valida si la clau primària existeix i veu que no.
4. El sistema avisa l'usuari que no té cap compte associat amb aquesta clau primària.

Extensió: Escenari codi introduït invàlid
6. El sistema valida que el codi sigui vàlid, però no ho és (sigui perquè ha caducat o perquè l'usuari s'ha equivocat). 7. El sistema avisa l'usuari que el codi no és vàlid i suggereix tornar a començar amb el procés de recuperació.

Usuaris: Gestió d'esdeveniments

Cas d'ús	**#7 Buscar esdeveniment per nom al buscador**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol buscar un esdeveniment per nom al buscador		
Escenari principal d'ús			
1. L'usuari accedeix a la vista d'esdeveniments. 2. L'usuari introdueix el nom de l'esdeveniment al buscador. 3. El sistema es comunica amb el servei de *Ticketmaster* i li demana les dades que ha introduït l'usuari. 4. El sistema de *Ticketmaster* retorna les dades sol·licitades. 5. El sistema li mostra els esdeveniments que coincideixen amb aquest nom.			
Extensió: no es troba l'esdeveniment			
4. El sistema de *Ticketmaster* no retorna esdeveniments. 5. El sistema indica l'usuari que no s'ha trobat cap esdeveniment amb aquest nom.			

Cas d'ús	**#8 Seguir un esdeveniment**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari no segueix ja aquest esdeveniment		
Disparador	-L'usuari vol seguir un esdeveniment		
Escenari principal d'ús			
1. L'usuari troba un esdeveniment en el qual està interessat. 2. L'usuari indica al sistema que vol seguir aquest esdeveniment. 3. El sistema retorna un codi d'acceptació.			

Cas d'ús	#9 Deixar de seguir un esdeveniment	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix aquest esdeveniment		
Disparador	-L'usuari vol deixar de seguir un esdeveniment		
Escenari principal d'ús			
1. L'usuari busca l'esdeveniment que vol deixar de seguir. 2. L'usuari indica al sistema que vol deixar de seguir aquest esdeveniment. 3. El sistema retorna un codi d'acceptació.			

Cas d'ús	#10 Llistar els esdeveniments que segueixes	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol veure un llistat de tots els esdeveniments que segueix		
Escenari principal d'ús			
1. L'usuari accedeix a la vista d'esdeveniments. 2. El sistema llista tots els esdeveniments que l'usuari està seguint.			

Cas d'ús	#11 Visualitzar un esdeveniment	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol veure un esdeveniment concret		
Escenari principal d'ús			
1. L'usuari accedeix a un esdeveniment concret. 2. El sistema li mostra la informació relativa a aquest esdeveniment.			

Usuaris: Gestió del mètode de recomanació

Cas d'ús	#12 Mostrar usuaris recomanats	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol veure les recomanacions d'altres perfils que li pugui fer el sistema		

Escenari principal d'ús
1. L'usuari accedeix a la funcionalitat de recomanació d'altres usuaris. 2. El sistema li va mostrant altres perfils amb els quals creu que podria ser afí depenent dels esdeveniments que els dos usuaris segueixin.

Cas d'ús	**#13 Enviar un missatge a un usuari que ens interessa**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment		
Disparador	-L'usuari està fent servir l'algoritme de recomanacions i aquest recomana un usuari que li interessa		
Escenari principal d'ús			
1. L'usuari està fent servir l'algoritme de recomanacions que li està mostrant usuaris. 2. L'usuari u1 troba un usuari u2 que li interessa i amb el que voldria tenir match. 3. L'usuari u1 vol enviar un missatge a l'usuari u2. U1 introdueix el text que vol enviar i li indica al sistema que l'enviï. 4. El sistema envia el missatge a l'usuari receptor. Fins que no hi hagi resposta, si n'hi ha, el xat no apareixerà a la finestra de xats de l'usuari emissor per a evitar que pugui enviar-li més missatges. 5. El sistema recomana un altre usuari o indica que de moment no queden més usuaris que mostrar.			

Cas d'ús	**#14 Rebutjar un usuari que no ens interessa**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment		
Disparador	-L'usuari està fent servir l'algoritme de recomanacions i aquest li recomana un usuari que no li interessa		
Escenari principal d'ús			
1. L'usuari està fent servir l'algoritme de recomanacions que li està mostrant usuaris. 2. El sistema recomana un usuari que no li interessa i amb el que no voldria tenir match. 3. L'usuari indica al sistema que no li interessa el perfil que li està mostrant. 4. El sistema recomana un altre usuari o indica que de moment no queden més usuaris que mostrar.			

Cas d'ús	**#15 Guardar un usuari que ens pot interessar**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment		

Disparador	-L'usuari està fent servir l'algoritme de recomanacions i aquest li recomana un usuari que li podria interessar
Escenari principal d'ús	
1. L'usuari està fent servir l'algoritme de recomanacions que li està mostrant usuaris. 2. L'usuari u1 troba un usuari u2 amb qui li podria interessar tenir match, però no vol enviar-li un missatge. 3. L'usuari u1 indica al sistema que guardi el perfil de l'usuari u2. Si l'usuari u2 envia un missatge de match o guarda a l'usuari u1, es generarà match i podran xatejar. Si el rebutja, no es genera match. 4. El sistema recomana un altre usuari o indica que de moment no queden més usuaris que mostrar.	

Cas d'ús	**#16 Navegar pel perfil d'un usuari**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment		
Disparador	-L'usuari està fent servir l'algoritme de recomanacions i vol veure tot el perfil d'un usuari recomanat		
Escenari principal d'ús			
1. L'usuari està fent servir l'algoritme de recomanacions que li està mostrant usuaris. 2. L'usuari pot navegar pel perfil de l'usuari que el sistema li ha recomanat.			

Cas d'ús	**#17 Respondre a un missatge de match**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment -L'usuari ha rebut un missatge de match		
Disparador	-L'usuari ha rebut un missatge de match i li interessa l'altre usuari		
Escenari principal d'ús			
1. L'usuari llegeix el missatge i veu el perfil de l'usuari emissor. 2. L'usuari respon al missatge perquè està interessat. 3. Es genera match.			

Cas d'ús	#18 Rebutjar un missatge de match	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment -L'usuari ha rebut un missatge de match		
Disparador	-L'usuari ha rebut un missatge de match i no li interessa l'altre usuari		
Escenari principal d'ús			
1. L'usuari indica al sistema que rebutja el match i aquest es cancel·la automàticament.			

Cas d'ús	#19 Llistar xats	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol veure tots els seus xats		
Escenari principal d'ús			
1. L'usuari accedeix a la finestra de xats. 2. El sistema li llista tots els xats que té.			

Cas d'ús	#20 Xatejar amb un usuari que ja és match	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment -Entre els dos usuaris hi ha match		
Disparador	-L'usuari vol xatejar amb un altre usuari amb qui té match		
Escenari principal d'ús			
1. L'usuari accedeix a la finestra de xats perquè vol iniciar una conversa o perquè ha rebut una notificació, ja que algun altre usuari li ha enviat un missatge. 2. L'usuari entra al xat que li interessa. 3. Els dos usuaris xategen			

Cas d'ús	#21 Desfer un match	Actor principal	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment -Entre els dos usuaris hi ha match		
Disparador	-L'usuari u1 ja no vol tenir match amb l'usuari u2		

Escenari principal d'ús
1. L'usuari accedeix a la finestra de xats on es llisten tots els matchs que té. 2. L'usuari busca el xat del match que vol desfer. 3. L'usuari entra en el xat i indica al sistema que vol desfer el match. 4. Els dos usuaris desapareixen de la llista de matchs de l'altre usuari, respectivament.

Cas d'ús	**#22 Reportar usuari**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema -L'usuari segueix algun esdeveniment -Entre els dos usuaris hi ha match		
Disparador	-L'usuari u1 vol reportar el comportament de l'usuari u2		
Escenari principal d'ús			
1. L'usuari accedeix a la finestra de xats on es llisten tots els matchs que té. 2. L'usuari busca el xat amb l'usuari que vol reportar. 3. L'usuari entra en el xat i indica al sistema que vol reportar aquest usuari. 4. El sistema indica que l'usuari ha estat reportat. 5. Els dos usuaris desapareixen de la llista de matchs de l'altre usuari, respectivament.			

Usuaris: Gestió del compte

Cas d'ús	**#23 Actualitzar les dades del nostre perfil**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol actualitzar les dades que mostra el seu perfil a altres usuaris		
Escenari principal d'ús			
1. L'usuari indica al sistema que vol editar el seu perfil. 2. El sistema li retorna les dades del seu perfil. 3. L'usuari actualitza les dades que li interessen del seu perfil i indica al sistema que guardi les noves dades. 4. El sistema actualitza les dades del perfil de l'usuari.			

Cas d'ús	**#24 Actualitzar les fotos del nostre perfil**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol actualitzar les fotografies que mostra al seu perfil a altres usuaris		

Escenari principal d'ús
1. L'usuari indica al sistema que vol editar el seu perfil. 2. El sistema li retorna les dades del seu perfil. 3. L'usuari selecciona del seu dispositiu la fotografia que li interessa. 4. L'usuari indica al sistema que actualitzi el seu perfil. 5. El sistema actualitza les fotografies del perfil de l'usuari.
Extensió: eliminar fotografia
2. L'usuari indica al sistema quina fotografia vol eliminar. 3. El sistema elimina la fotografia del seu perfil.

Cas d'ús	**#25 Modificar la configuració del compte**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol actualitzar la seva configuració (contrasenya i gènere) del compte		
Escenari principal d'ús			
1. L'usuari accedeix a l'apartat de configuració del compte. 2. El sistema li retorna la configuració del seu perfil. 3. L'usuari actualitza els camps en els que estigui interessat. 4. L'usuari indica al sistema que guardi la configuració. 5. El sistema valida els canvis. 6. El sistema actualitza la configuració.			
Extensió: canvi invàlid			
5. El sistema veu que algun d'aquests canvis és invàlid. 6. El sistema avisa l'usuari de quin dels canvis és invàlid. 7. Si l'usuari surt de la finestra sense haver guardat i el sistema validat els nous canvis, aquests no s'apliquen i es mantenen els que ja estaven.			

Cas d'ús	**#26 Modificar les preferències de l'algoritme de recomanació**	**Actor principal**	Usuari
Precondicions	-L'usuari està loguejat al sistema		
Disparador	-L'usuari vol modificar la configuració de l'algoritme de recomanació		
Escenari principal d'ús			

1. L'usuari accedeix a l'apartat de configuració de recomanació.
2. El sistema li retorna la configuració de recomanació del seu compte.
3. L'usuari actualitza els camps en què estigui interessat i indica al sistema que guardi la configuració.
4. El sistema valida els canvis i actualitza la configuració.

6.3. Model conceptual

El model conceptual és la representació dels conceptes (objectes) significatius en el domini del problema. Mostra, principalment, classes d'objectes, associacions entre classes d'objectes, atributs de les classes d'objectes i restriccions d'integritat, gràfiques i textuals [29]. Un objecte és una entitat que existeix al món real. Té identitat pròpia i és distingible dels altres objectes.

6.3.1. Esquema conceptual de les dades

L'esquema conceptual identifica les relacions a alt nivell entre les diferents entitats del nostre sistema. Mostra els seus atributs, els tipus d'aquests i les relacions que es formen entre les diferents entitats.

Les entitats més importants del sistema són la classe Usuari i la classe Esdeveniment. A partir d'aquestes dues, es construeixen les altres i s'estableixen les dependències entre elles.

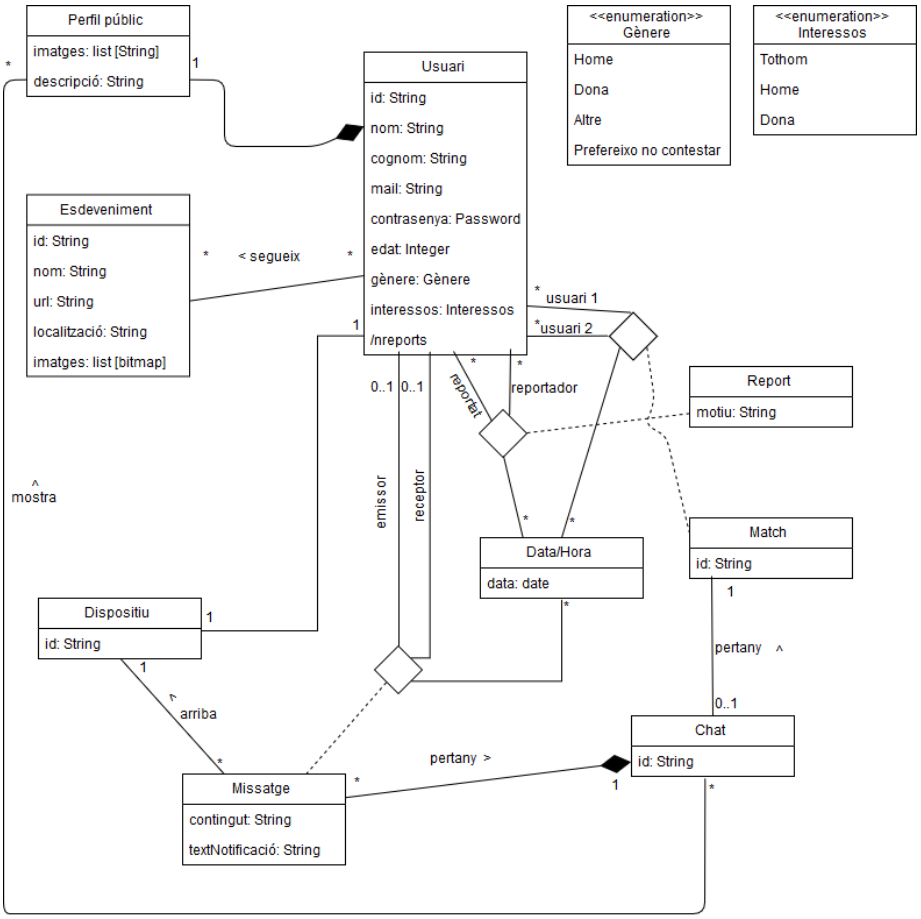


Figura 5: esquema conceptual de dades del sistema

6.3.2. Restriccions d'integritat

Les restriccions d'integritat són aquelles condicions que no es poden representar a l'esquema conceptual, però que també cal tenir en compte per tal d'assegurar que el sistema funciona tal i com s'ha dissenyat.

- Claus externes: (Usuari, id), (Esdeveniment, id), (Data/Hora, date), (Dispositiu, id)
- El camp /nreport d'usuari estàndard es calcula a partir del nombre de vegades que surt usuari a l'associació Report.
- Un usuari no pot tenir més de 3 reports.
- Dos usuaris poden tenir match si segueixen el mateix esdeveniment.
- Un usuari només pot reportar a usuaris amb els qui té match.
- Una notificació només arribarà al dispositiu del receptor del missatge.
- Un usuari no pot tenir dos matchs a la vegada amb el mateix usuari.
- Un usuari no pot tenir match amb ell mateix.
- Els usuaris d'un missatge de xat han de ser els mateixos que els que té el match del xat.

6.3.3. Descripció de les classes

Usuari
Representa el grup d'usuaris que interactuaran amb el sistema.

- id: identificador de cada usuari per a usos interns del sistema.
- nom: nom de l'usuari. Es mostrarà en el seu perfil i de cara a altres usuaris.
- cognom: cognom de l'usuari. Com el nom, també es mostrarà al seu perfil i el podran veure altres usuaris.
- mail: correu electrònic de l'usuari. Serveix per a accedir al seu compte i per a recuperar la contrasenya. No serà públic a altres usuaris.
- contrasenya: clau privada per a accedir al compte.
- edat: edat de l'usuari.
- gènere: gènere de l'usuari. És un dels filtres aplicables a l'algoritme de recomanació.
- interessos: interessos de l'usuari. Determina el gènere de les recomanacions que li mostrarà l'algoritme de recomanació.
- /nreports: nombre de vegades que l'usuari ha estat reportat. El màxim és 3, després d'això s'elimina l'usuari del sistema. Es calcula a partir del nombre de vegades que surt usuari a l'associació Report.

Perfil públic
Cada usuari estàndard té associat un perfil públic que es mostra als altres usuaris de la plataforma.

- imatges: imatge o imatges pujades per l'usuari que es mostraran públicament.
- descripció: petita descripció de l'usuari respecte ell/a mateix/a. Es mostra de cara al públic.

Esdeveniments
Element principal del sistema. Representa un esdeveniment que els usuaris poden seguir per informar que estan interessats en assistir.

- id: pot ser genèrica o pròpia. La genèrica ve assignada per *Ticketmaster* i la pròpia la genera el sistema per a ajudar a identificar més fàcilment les instàncies.
- nom: nom de l'esdeveniment.
- url: enllaç a alguna pàgina amb més informació relativa a l'esdeveniment.
- localització: localització on tindrà lloc l'esdeveniment.

- imatges: conjunt d'imatges relatives a l'esdeveniment.

Report
Representa els reports que ha rebut un usuari en concret i el motiu d'aquest. El compte d'un usuari és eliminat del sistema si rep més de 3 reports. Es guarda l'hora, el reportat i el reportador.

- motiu: motiu o justificació de per què el report ha estat generat.

Match
Representa la unió entre dos usuaris. Quan els usuaris s'accepten entre ells, es genera match. Un match permet xatejar sense cap mena de restricció.

- id: id del match per a ús intern i per facilitar l'enviament de les notificacions.

Chat
Representa un conjunt de missatges entre dos usuaris que tenen match. Es pot accedir al perfil del l'altre usuari.

- id: id del chat per a ús intern i per facilitar l'enviament de les notificacions.

Missatge
Dins del context del xat, un usuari pot decidir enviar un missatge a un altre. Es guarda l'emissor, el receptor i l'hora de l'enviament.

- contingut: contingut del missatge.
- textNotificació: contingut de la notificació. Podrà ser el contingut del missatge o un text genèric convidant l'usuari a accedir a l'aplicació per a poder veure el contingut amagat.

Dispositiu
Representa el dispositiu on l'usuari farà servir l'aplicació i on s'enviaran les notificacions.

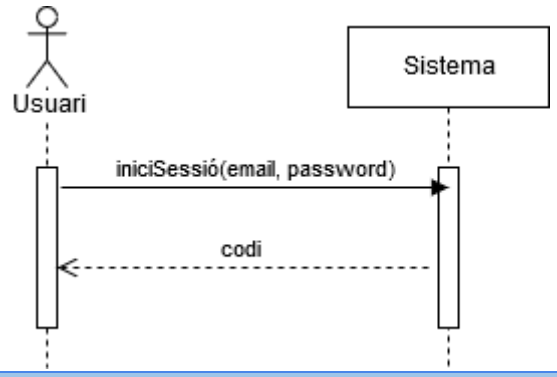
- id: id del dispositiu per a ús intern i per facilitar l'enviament de les notificacions.

6.4. Model de comportament

En orientació a objectes, aquests es comuniquen mitjançant la invocació d'operacions d'altres objectes. Els models de comportament mostren la seqüència d'esdeveniments entre els actors i el sistema i permeten identificar i mostrar l'efecte de les operacions del sistema.

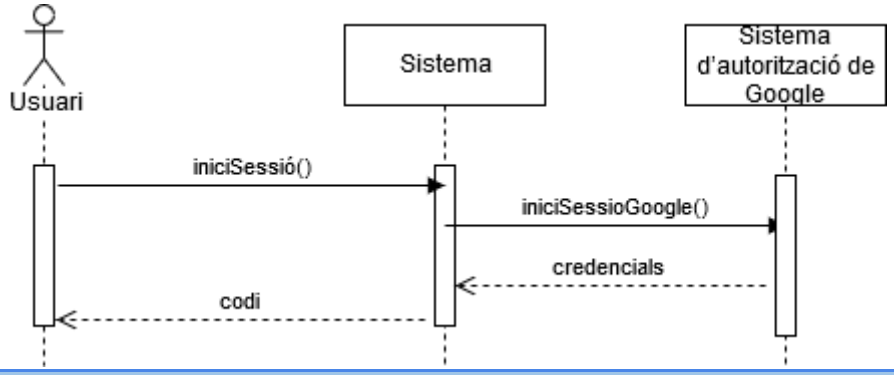
En aquest apartat, per cada cas d'ús es mostra la seqüència d'operacions que es cridaran. De cada crida, s'especifica la seva precondició (aspectes que s'han de complir abans d'executar l'operació per assegurar el seu correcte funcionament), la postcondició (aspectes que es garanteixen un cop executada l'operació) i el seu body (què retorna la crida).

Log In



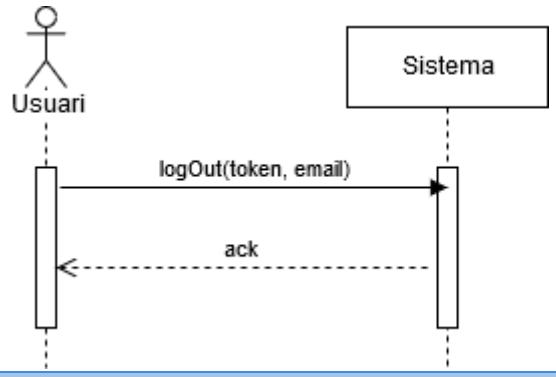
iniciSessió(email:String, password:Password)	
Precondició:	-
Postcondició:	-
Body:	-Es retorna un token d'accés o un codi d'error si les credencials no són vàlides.

Log in amb *Google*



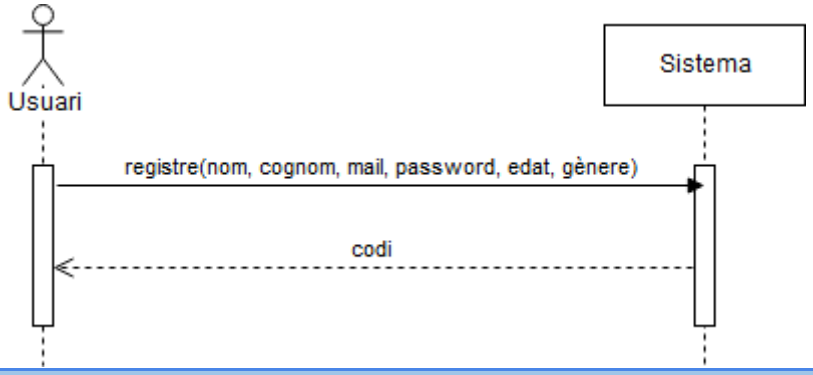
iniciSessió()	
Precondició:	-
Postcondició:	-El sistema invoca l'execució de l'operació iniciSessioGoogle() del sistema d'autorització de *Google*.
Body:	-Es retorna un token d'accés o un codi d'error depenent de la resposta del servei de *Google*.

Log out



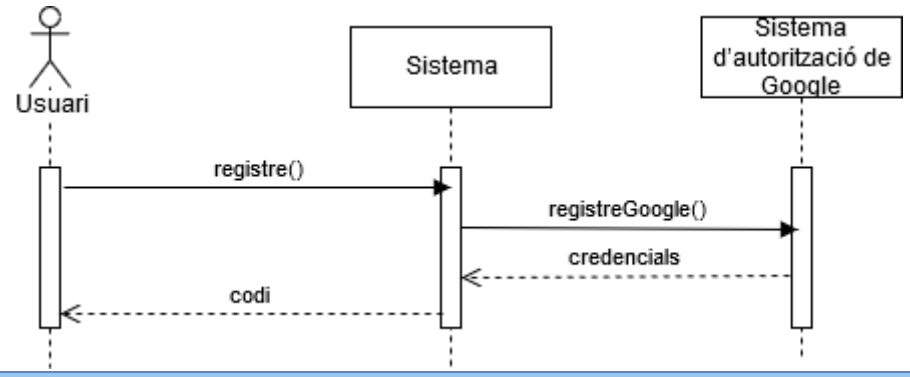
logOut(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-El *token* deixa de ser vàlid i es crearà un de nou en el següent log in.
Body:	-

Registre



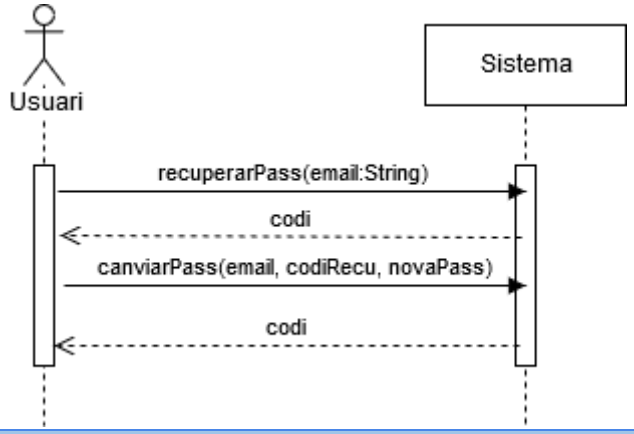
registre(nom:String, cognom:String, mail:String, password:Password, edat: int, gènere:Gènere)	
Precondició:	-
Postcondició:	-Es crea un usuari estàndard amb les dades introduïdes.
Body:	-Es retorna un codi d'acceptació si s'ha creat la instància correctament o un codi d'error altrament.

Registre amb *Google*



registre()	
Precondició:	-
Postcondició:	-El sistema invoca l'execució de l'operació registreGoogle() del sistema d'autorització de *Google*. -Es crea un usuari estàndard amb les credencials rebudes del sistema de *Google*.
Body:	-Es retorna un codi d'acceptació si la instància d'usuari s'ha pogut crear correctament o un codi d'error altrament.

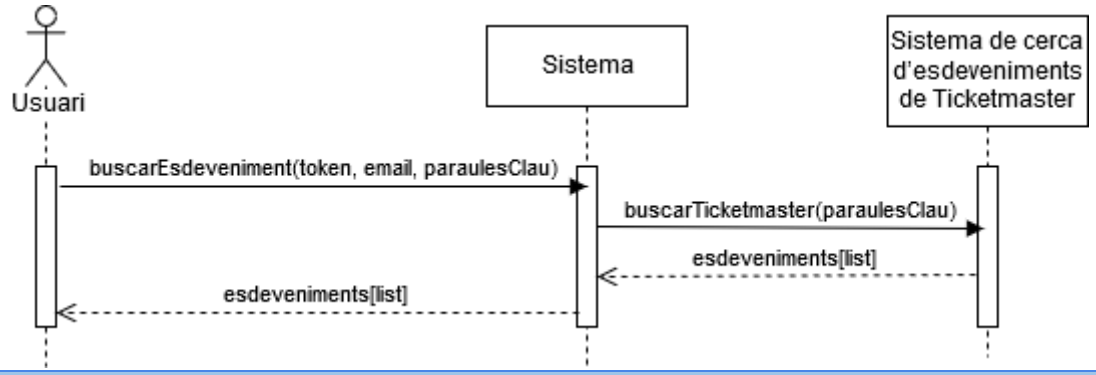
Recuperació de contrasenya



recuperarPass(email:String)	
Precondició:	-
Postcondició:	-S'envia el codi de Recuperació al *email* si aquest consta al sistema.
Body:	-Es retorna un codi d'acceptació si s'ha enviat el mail o un codi d'error si el *email* no consta al sistema.

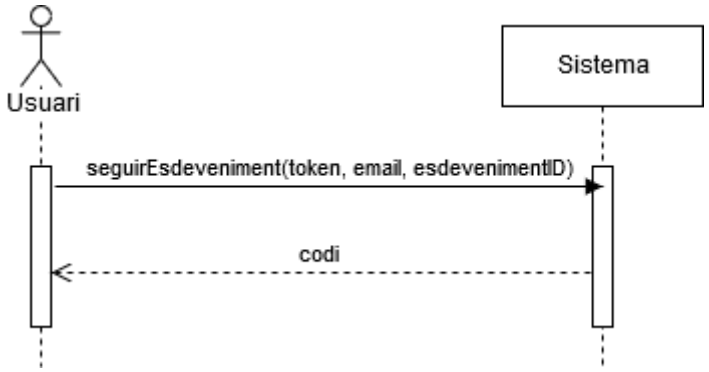
canviarPass(email:String, codiRecu:String, novaPass:Password)	
Precondició:	-
Postcondició:	-Es canvia la contrasenya del compte de l'usuari per *novaPass*.
Body:	-Es retorna un codi d'acceptació si s'ha realitzat el canvi de contrasenya o codi d'error altrament.

Buscar esdeveniment



buscarEsdeveniment(token:String, email: String, paraulesClau:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-El sistema invoca l'execució de l'operació buscarTicketmaster() del sistema de cerca d'esdeveniments de *Ticketmaster* i li passa les paraules clau.
Body:	-Es retorna la llista de [0..*] esdeveniments que retorna l'operació buscarTicketmaster().

Seguir esdeveniment



seguirEsdeveniment(token:String, email:String, esdevenimentID:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*esdevenimentID* existeix.
Postcondició:	-Es crea l'associació entre l'usuari i l'esdeveniment *esdevenimentID*.
Body:	-Es retorna un codi d'acceptació si s'ha creat l'associació entre l'usuari i l'esdeveniment *esdevenimentID* o un codi d'error altrament.

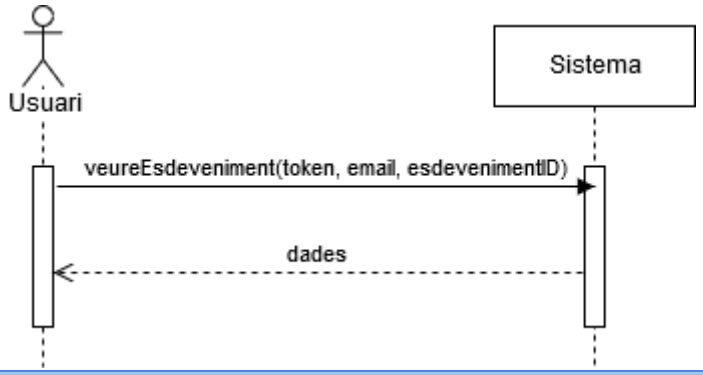
Deixar de seguir esdeveniment

deixarDeSeguirEsdeveniment(token:String, email:String, esdevenimentID:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*esdevenimentID* existeix. -L'usuari segueix l'esdeveniment *esdevenimentID*.
Postcondició:	-Es desfà l'associació entre l'usuari i l'esdeveniment *esdevenimentID*.
Body:	-Es retorna un codi d'acceptació si s'ha desfet l'associació entre l'usuari i l'esdeveniment *esdevenimentID* o un codi d'error altrament.

Llistar els esdeveniments que segueixes

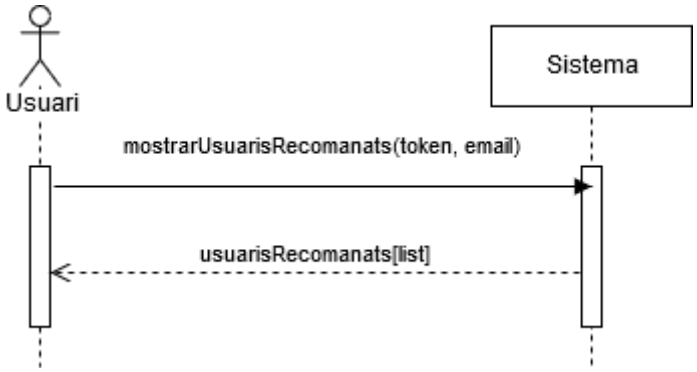
llistarEsdeveniment(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-
Body:	-Es retorna una llista [0..*] amb els esdeveniments que segueix l'usuari.

Visualitzar un esdeveniment



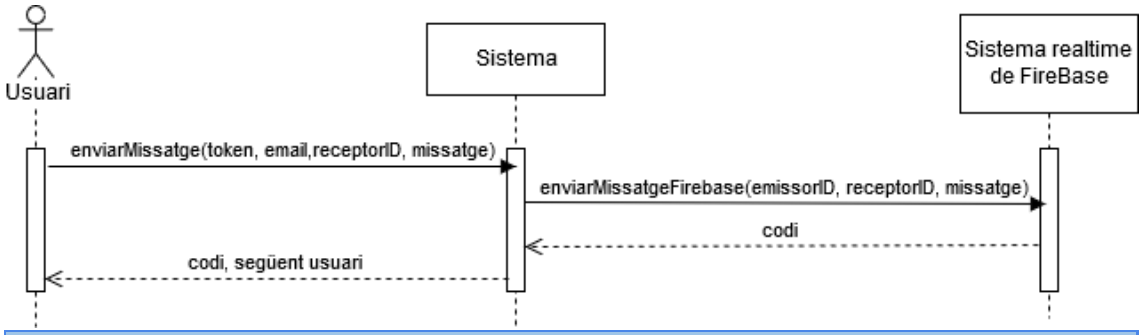
veureEsdeveniment(token:String, email:String, esdevenimentID:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*esdevenimentID* existeix al sistema.
Postcondició:	-
Body:	-Es retorna la informació relativa a l'esdeveniment *esdevenimentID*.

Mostrar usuaris recomanats



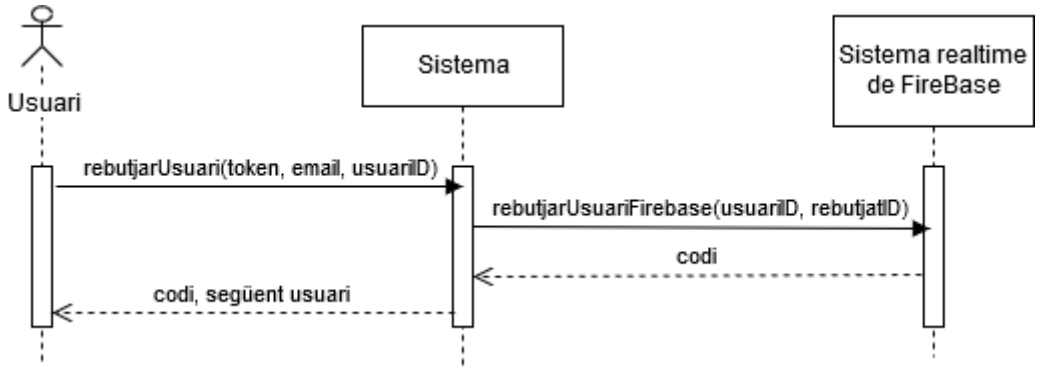
mostrarUsuarisRecomanats(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-
Body:	-Es retorna una llista amb els usuaris recomanats pel sistema.

Enviar un missatge a un usuari que ens interessa



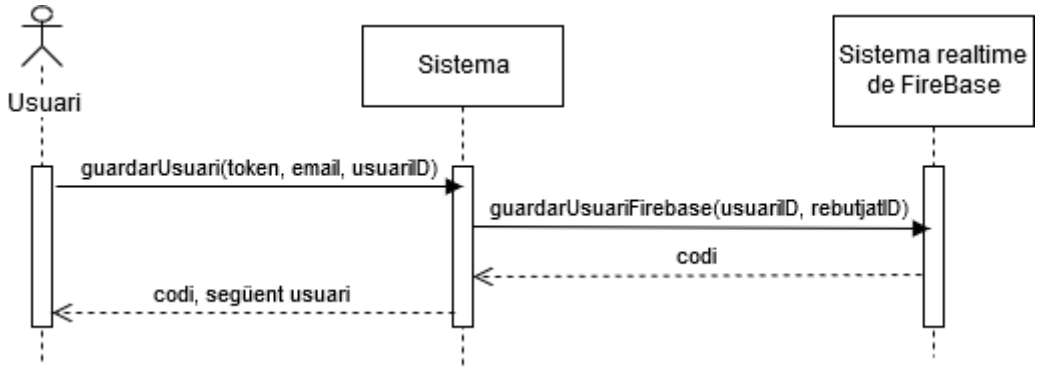
enviarMissatge(token:String, email:String, receptorID:String, missatge:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*receptorID* existeix al sistema.
Postcondició:	-El sistema invoca l'execució de l'operació enviarMissatgeFirebase() del sistema realtime de *FireBase* i li passa els ids de l'emissor, el receptor i el missatge. L'operació envia el *missatge* a l'usuari *receptorID*.
Body:	-Es retorna un codi d'acceptació o error depenent de la resposta de *Firebase* juntament amb el següent perfil a recomanar.

Rebutjar un usuari que no ens interessa



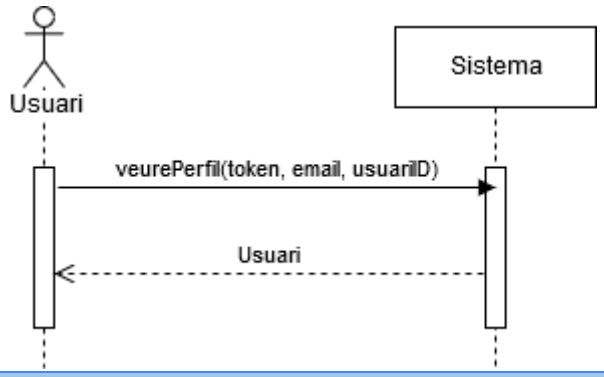
rebutjarUsuari(token:String, email:String, usuariID:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*usuariID* existeix al sistema.
Postcondició:	-El sistema invoca l'execució de l'operació rebutjarUsuariFirebase() del sistema realtime de *FireBase* i li passa *l'usuariID* i l'id de l'usuari rebutjat *rebutjatID*.
Body:	-Es retorna un codi d'acceptació o error depenent de la resposta de *Firebase* juntament amb el següent perfil a recomanar.

Guardar un usuari que ens pot interessar



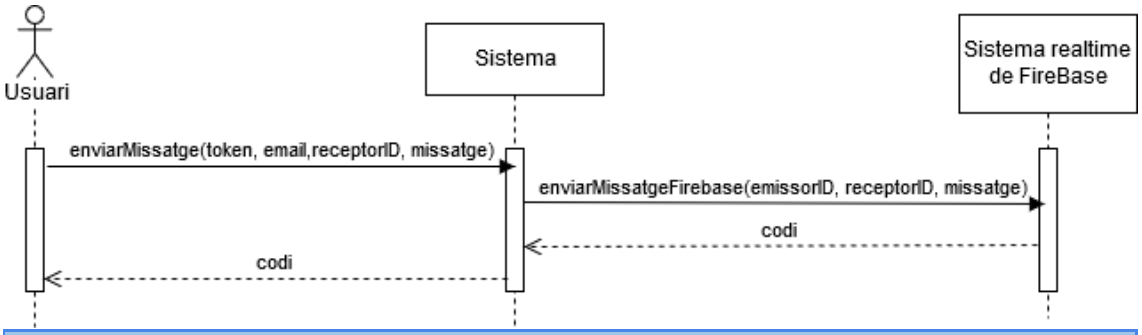
guardarUsuari(token:String, email:String, usuariID:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*usuariID* existeix al sistema.
Postcondició:	-El sistema invoca l'execució de l'operació guardarUsuariFirebase() del sistema realtime de *FireBase* i li passa *l'usuariID* i l'id de l'usuari rebutjat *rebutjatID*.
Body:	-Es retorna un codi d'acceptació o error depenent de la resposta de *Firebase* juntament amb el següent perfil a recomanar.

Navegar pel perfil d'un usuari



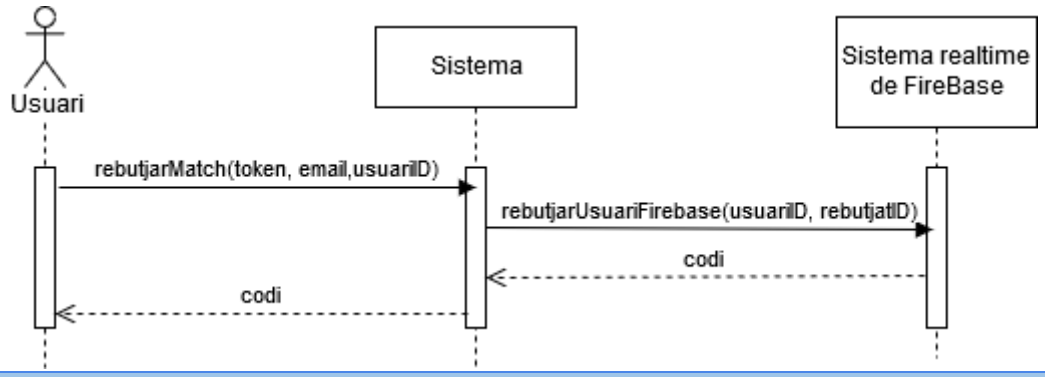
veurePerfil(token:String, email:String, usuariID:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*usuariID* existeix al sistema.
Postcondició:	-
Body:	-Es retorna el perfil de l'usuari *usuariID* o un codi d'error, altrament.

Respondre a un missatge de match



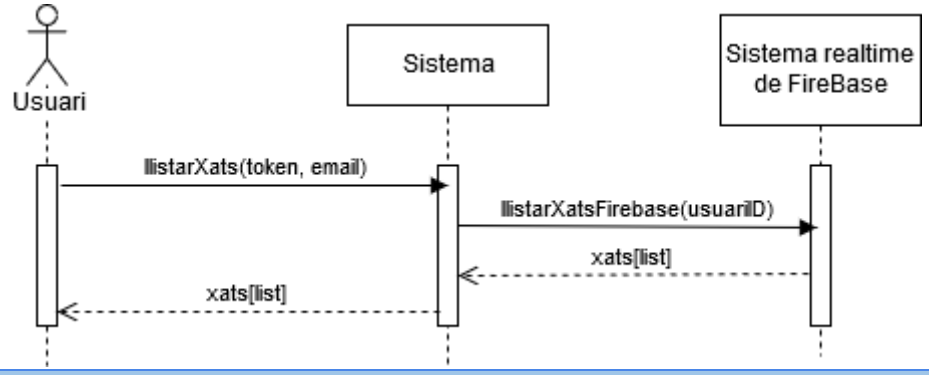
enviarMissatge(token:String, email:String,receptorID:String, missatge:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*usuariID* existeix al sistema.
Postcondició:	-El sistema invoca l'execució de l'operació enviarMissatgeFirebase() del sistema realtime de *FireBase* i li passa els ids de l'emissor el receptor i el missatge. L'operació envia el missatge a l'usuari *receptorID*.
Body:	-Es retorna un codi d'acceptació o error depenent de la resposta de *Firebase*.

Rebutjar un missatge de match



rebutjarMatch(token:String, email:String,usuariID:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*usuariID* existeix al sistema.
Postcondició:	-El sistema invoca l'execució de l'operació rebutjarMatchFirebase() del sistema realtime de *FireBase* i li passa l'ID de l'usuari i de l'usuari rebutjat.
Body:	-Es retorna un codi d'acceptació o error depenent de la resposta de *Firebase*.

Llistar xats

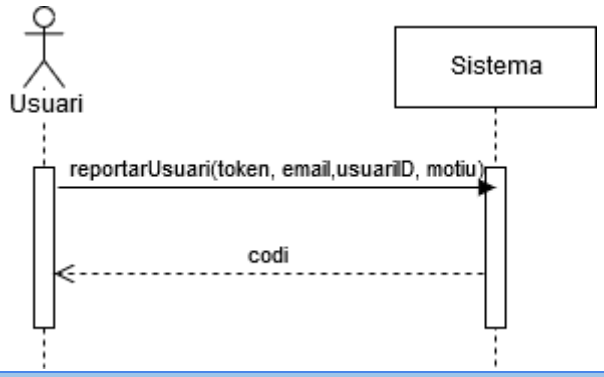


llistarXats(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-El sistema invoca l'execució de l'operació llistarXatsFirebase() del sistema realtime de *FireBase* i li passa l'ID de l'usuari del qual vol llistar els xats.
Body:	-Es retorna un llistat de xats.

Xatejar amb un usuari que ja és match

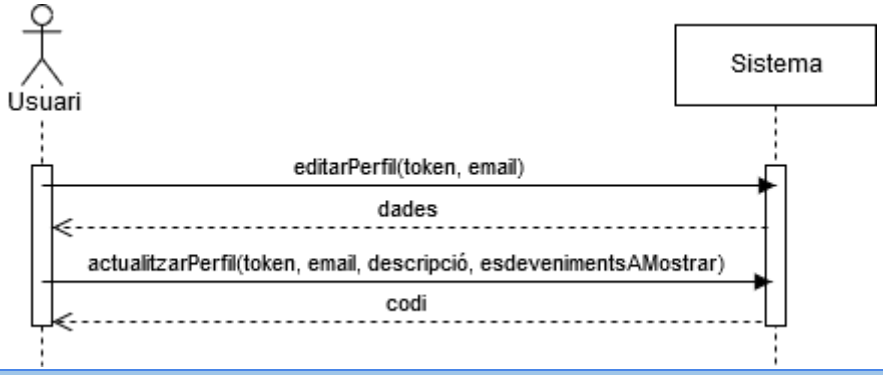
Comportament idèntic a Respondre a un missatge de match en loop.

Reportar usuari



reportarUsuari(token:String, email:String,usuariID:String, motiu:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -*usuariID* existeix al sistema.
Postcondició:	-El sistema genera una instància de *Report*.
Body:	-Es retorna un codi d'acceptació si s'ha creat l'instancia de Report o d'error altrament.

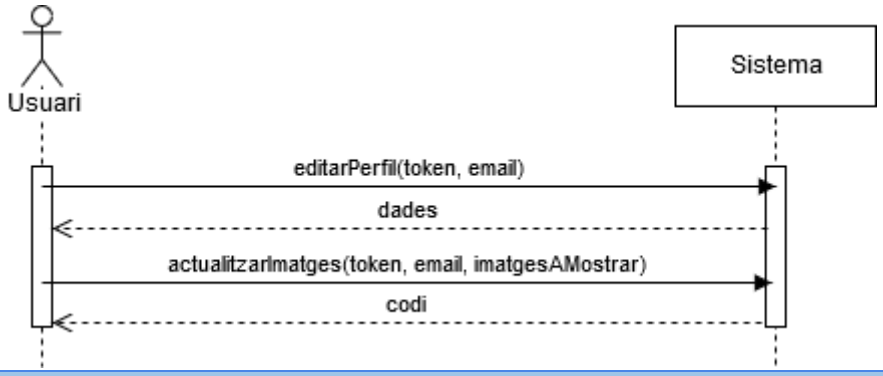
Actualitzar les dades del nostre perfil



editarPerfil(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-
Body:	-Es retornen les dades de l'usuari.

actualitzarPerfil(token:String, email:String, descripció:String, esdevenimentsAMostrar:list[esdevenimentsID:String])	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -les dades tenen un format vàlid.
Postcondició:	-S'actualitzen les dades de l'usuari.
Body:	-Es retorna un codi d'acceptació si s'han actualitzat les dades o un codi d'error altrament.

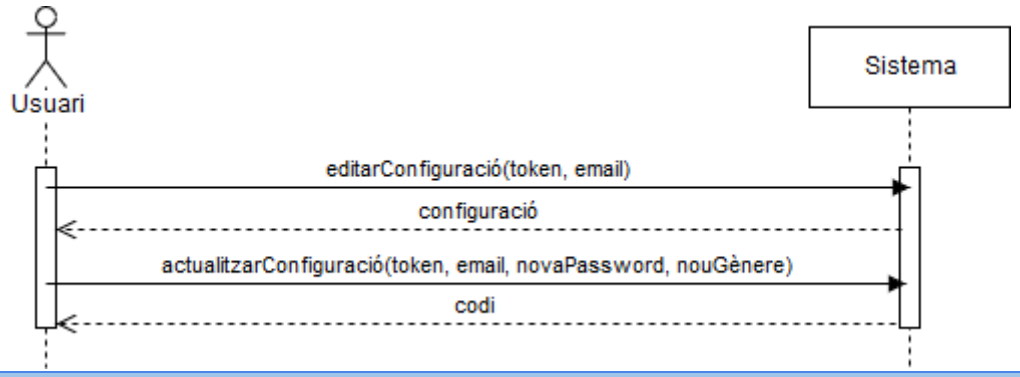
Actualitzar les fotos del nostre perfil



editarPerfil(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-
Body:	-Es retornen les dades de l'usuari.

actualitzarImatges(token:String, email:String, imatgesAMostrar:list[imatges:bitmap])	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -les imatges tenen un format vàlid.
Postcondició:	-S'actualitzen les imatges de l'usuari.
Body:	-Es retorna un codi d'acceptació si s'han actualitzat les imatges o un codi d'error altrament.

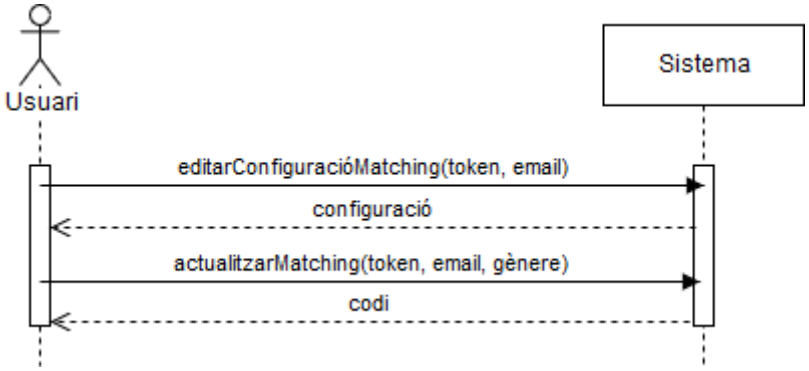
Modificar la configuració del compte



editarConfiguració(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-
Body:	-Es retornen la configuració actual de l'usuari.

actualitzarConfiguració(token:String, email:String, novaPassword: Password, nouGènere: String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -les dades tenen un format vàlid.
Postcondició:	-S'actualitza la configuració de l'usuari.
Body:	-Es retorna un codi d'acceptació si s'ha actualitzat la configuració o un codi d'error altrament.

Modificar les preferències de l'algoritme de recomanació



editarConfiguracióMatching(token:String, email:String)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema.
Postcondició:	-
Body:	-Es retorna la configuració de l'algoritme de recomanació de l'usuari.

actualitzarMatching(token:String, email:String, gènere: Gènere)	
Precondició:	-El *token* és vàlid. -*email* existeix al sistema. -les dades tenen un format vàlid.
Postcondició:	-S'actualitzen les preferències de l'algoritme de recomanació de l'usuari.
Body:	-Es retorna un codi d'acceptació si s'han actualitzat preferències de l'algoritme de recomanació o un codi d'error altrament.

7. Disseny

Un cop s'ha realitzat l'anàlisi de requisits, s'han entès les exigències del client i s'ha descrit el comportament extern del sistema des del punt de vista de l'usuari i de l'entorn, es procedeix a definir el disseny del sistema software. Aquest consistirà en determinar els components que seran necessaris per a garantir el correcte funcionament de les funcionalitats requerides.

L'objectiu d'aquest capítol és definir l'arquitectura que se seguirà per al desenvolupament del sistema per construir. Es faran servir diversos artefactes per tal de fer més visual l'explicació.

7.1. Arquitectura lògica

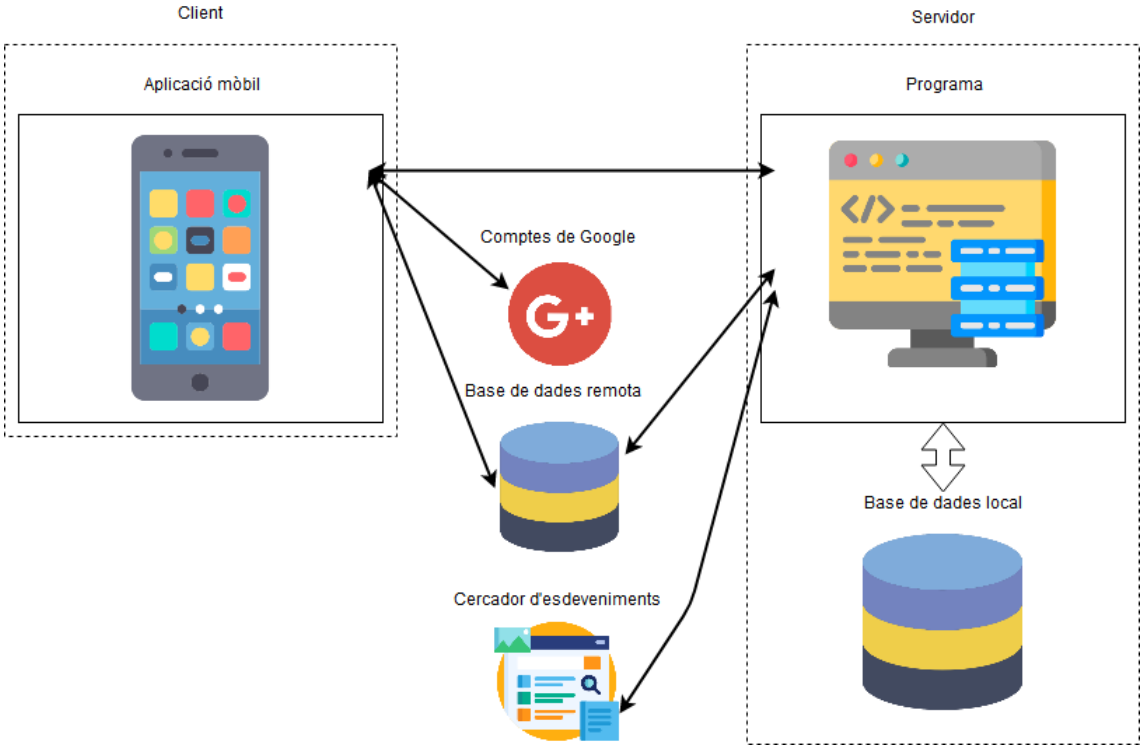


Figura 6: Abstracció de l'arquitectura lògica del sistema

El sistema per desenvolupar estarà format per dos components clau mostrats a la figura 6. Per la banda del client, una aplicació mòbil nativa (d'ara en endavant, app), és a dir, amb el llenguatge propi del sistema triat, que s'encarregarà d'interactuar amb l'usuari i mostrar les dades que retornarà el servidor. Per altra banda, un servidor per al tractament del back end que actuarà com API i rebrà totes les peticions del client, les processarà i retornarà el resultat pertinent. Aquest servidor també interaccionarà amb els serveis externs d'emmagatzematge en temps real i el cercador d'esdeveniments.

L'app mòbil, tot i que s'ha pensat per a un entorn Android, es pot abstreure a qualsevol mena d'arquitectura que permeti aplicar el paradigma de l'orientació a objectes. L'app estarà dissenyada seguint una arquitectura per capes. Tal com indica Mark Richards [30], "Els components del patró d'arquitectura en capes s'organitzen en capes horitzontals, cada capa exercint un paper específic dins de l'aplicació". En el nostre cas es faran servir les quatre capes de la figura 7: presentació, domini del front end, persistència de dades del front end i la responsable de la comunicació amb el servidor i els serveis externs.

Encara que la gestió del domini del sistema es realitza al servidor, a l'app també caldrà una mínima lògica, atès que es necessita gestionar la sessió de l'usuari en qüestió i guardar algunes dades en memòria local per evitar peticions contínues. A més, el sistema de xat es gestiona gairebé íntegrament a l'app amb mínima interacció del servidor.

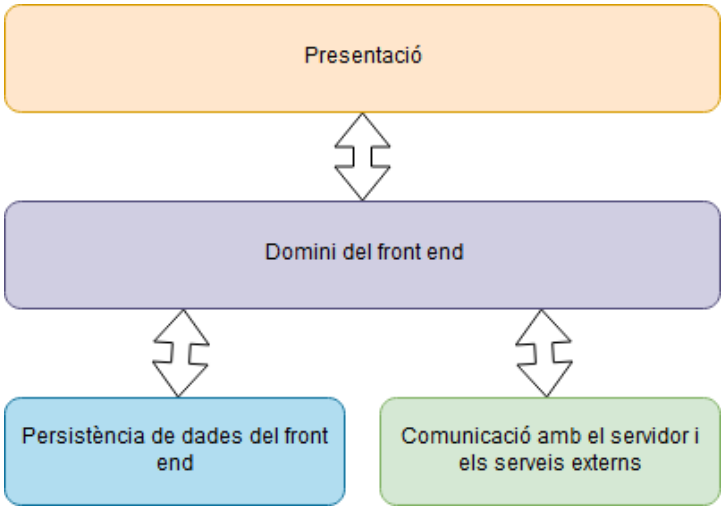


Figura 7:Diagrama de les capes de l'app (component front end)

Per altra banda, ja que es vol potenciar la reutilització de codi i la separació entre components, s'aplicarà el patró MVC (model-vista-controlador). Aquest es basa en separar la lògica d'una aplicació de les seves vistes, o el que és el mateix, separar el funcionament de la representació que veu l'usuari. El patró MVC estàndard, i també el que es farà servir en aquest projecte, consta de tres components. Codeacademy [31] els defineix de la següent forma:

- Model: És la representació de la informació amb la qual el sistema opera, per tant gestiona tots els accessos a aquesta informació, tant consultes com actualitzacions.
- Vista: El codi de visualització es compon de totes les funcions que interactuen directament amb l'usuari. Aquest és el codi que determina com es veu l'aplicació i a la vegada, defineix la interacció amb l'usuari.
- Controlador: El codi del controlador actua com a enllaç entre el Model i la Vista, rebent les dades de l'usuari i decidint què fer amb elles. És el cervell de l'aplicació i uneix el model i la vista.

Per tant, en el cas de la nostra app, l'usuari interaccionarà amb la pantalla (vista) i l'acció que prengui serà interceptada pel controlador, qui, si és necessari, demanaria les dades pertinents al model o a altres controladors per a després inflar la vista de nou amb el contingut actualitzat.

Un exemple d'aquest comportament es pot observar en la figura 8. Fusionant aquest amb el diagrama de la figura 7, les vistes i controladors estarien dins la capa de presentació i el model inclouria la resta de capes: domini del front end, persistència de dades del front end i la comunicació amb servidor i serveis externs.

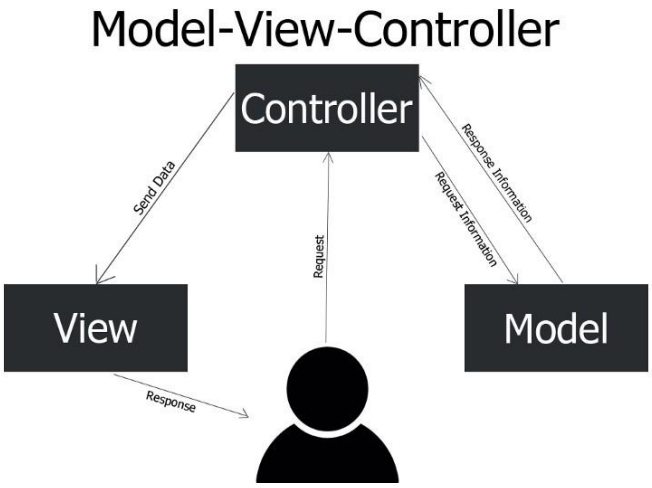


Figura 8: Model-Vista-Controlador. Font: medium.com

L'altre component del sistema és el servidor encarregat de la gestió del back end. Aquest component funcionarà com a API entre l'app mòbil i el tractament de les dades. Una API és un conjunt d'interfícies, i tal com defineix ServiceTonic[32], "una «interfície» és la forma en què dues aplicacions o serveis es comuniquen entre si. Ho fan exposant a la resta d'aplicacions el conjunt de serveis disponibles a cadascuna i com s'han d'accedir-hi."

Dintre del servidor, les funcionalitats seran agrupades en paquets depenent del model sobre el què actuen, ajudant així a estructurar el sistema d'una forma lògica i agilitzant el procés de desenvolupament, debugging i testing. Cada paquet estarà associat a un model, que serà l'objecte sobre qui es construiran les funcionalitats. Dins de cada paquet hi haurà 4 entitats diferenciades:

- Controladors: funcionaran igual que en el patró MVC aplicat a l'app. Cada crida permesa per l'API estarà associada amb una funció, que s'encarregarà de realitzar les crides a altres controladors o delegar la feina en els repositoris.
- Repositori: S'encarregaran principalment de fer el tractament de les dades i de funcionalitats bàsiques amb aquestes. Interactuaran amb el model de les dades i amb els mappers per a guardar i llegir la base de dades.
- Models: igual que en el patró MVC, són la representació de la informació amb la qual el sistema opera, per tant gestionen tots els accessos a aquesta informació, tant consultes com actualitzacions.
- Mappers: S'encarregaran de transformar les dades de la base de dades en objectes del model de forma automàtica, facilitant la integritat de la informació emmagatzemada al sistema.

A més, el servidor haurà de tractar amb serveis externs com el cercador d'esdeveniments i la base de dades en temps real. En aquest cas no es crearà una capa específica per a realitzar el tractament de les dades ja que cada servei en requereix un de diferent i no es pot abstreure a un codi comú. Per tant, els repositoris que necessitin realitzar una crida externa la processaran ells mateixos.

Com a imatge global, quan l'usuari realitzi una acció en l'app, aquesta serà capturada pel controlador de la vista, que es comunicarà amb el model de les dades o amb altres controladors. Si l'acció sol·licitada per l'usuari necessita un tractament de les dades, és molt probable que

s'hagi de fer una crida al servidor. Per fer-la, l'app haurà d'enviar la informació requerida, en el format requerit a la ruta on el servidor estigui escoltant. Quan el servidor rebi la consulta (en argot informàtic *query*) aquesta serà capturada pel controlador encarregat de la ruta especificada. Aquest controlador, a partir de la funcionalitat que estigui realitzant, es comunicarà amb el seu repositori, un altre controlador o un model de dades. Si ho fa amb el seu repositori, dins d'aquest es podrà fer, si cal, la petició a un dels serveis externs o fer el tractament de les dades pròpiament. Si aquestes estan guardades a la base de dades local, caldrà passar pels mappers. Un cop es tingui la informació i s'hagin aplicat les modificacions pertinents, es retornarà a l'app el resultat, que serà inflat de nou a la vista que veu l'usuari per part del controlador d'aquesta.

Es mostra un resum de l'arquitectura lògica del sistema a la figura 9.

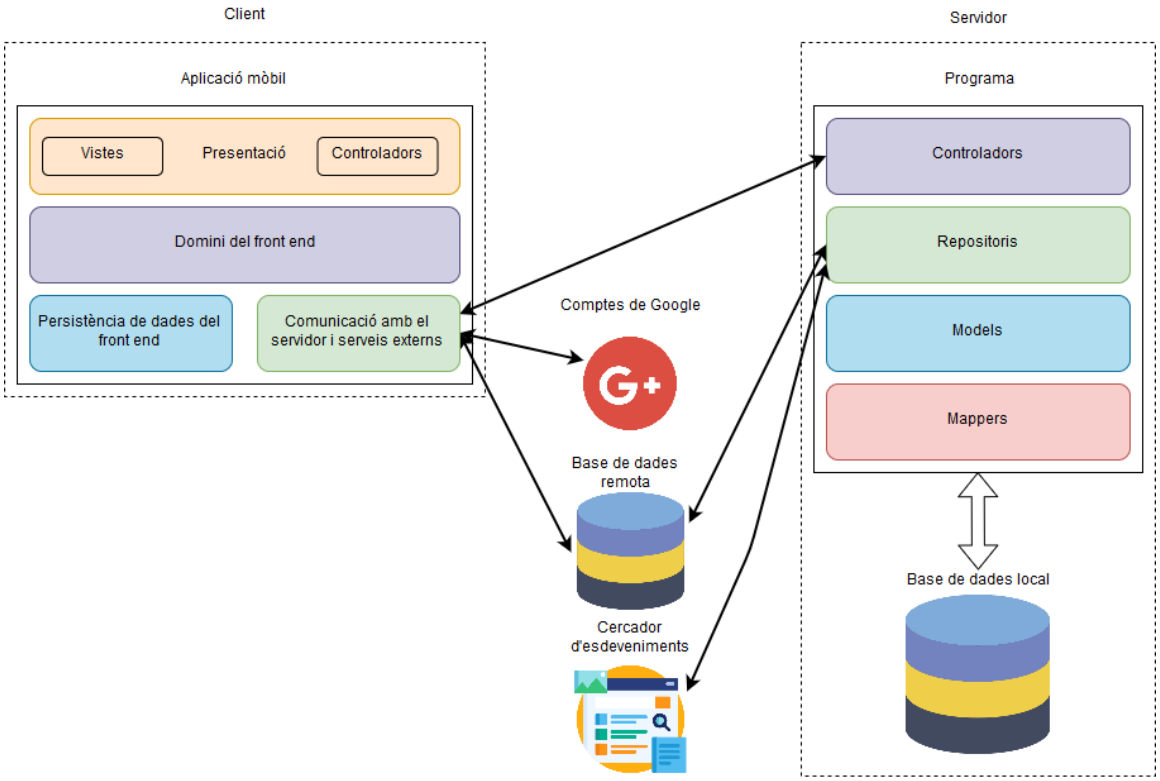


Figura 9: Arquitectura lògica del sistema

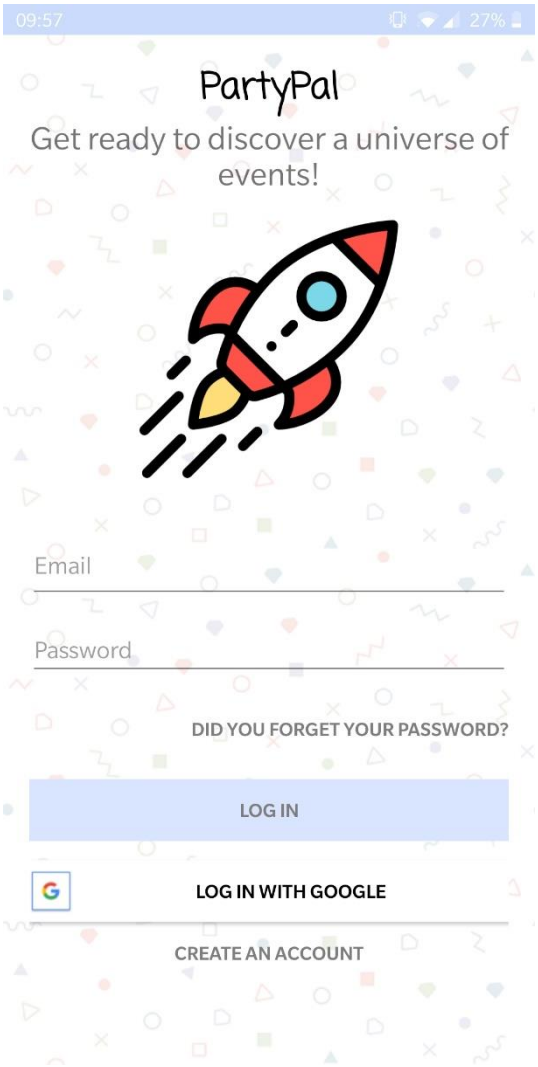
7.2. Disseny de l'app Android (front end)

Abans d'entrar plenament en el detall de l'arquitectura s'ha cregut convenient mostrar com es veuria la interfície d'usuari i com es navegaria per l'app.

7.2.1. Disseny extern de la capa de presentació

Els següent apartat recull els *mockups* o esborranys de les vistes amb les quals interaccionaria l'usuari, acompanyades del nom intern que se li donarà i una breu explicació de les funcionalitats que inclouran.

Inici de sessió



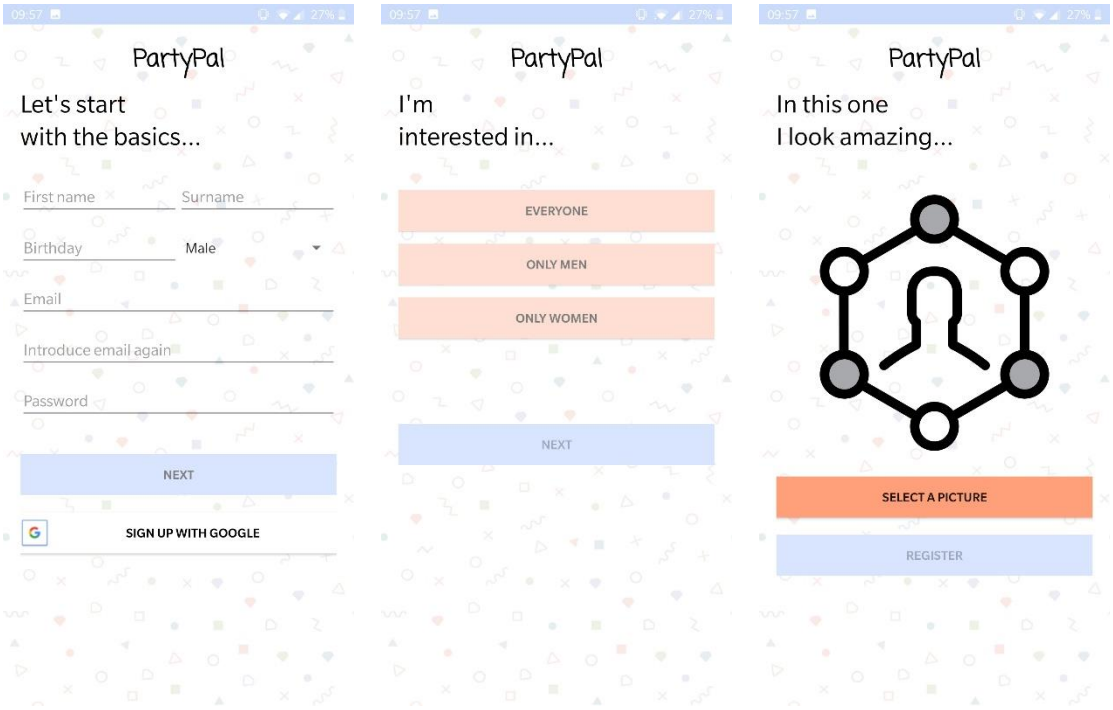
Primera vista que veurà l'usuari en accedir a l'app. Permet introduir el correu electrònic especificat al registre i la contrasenya actual per tal d'iniciar la sessió i entrar al sistema.

Aquesta vista també permetrà fer l'inici de sessió amb *Google*.

Des d'aquesta vista també es podrà accedir a la funcionalitat de recuperació de contrasenya i al registre d'usuari.

Aquesta vista s'anomenarà login.xml.

Registre



D'esquerra a dreta: registreDades.xml, registreInteressos.xml i registreFoto.xml.

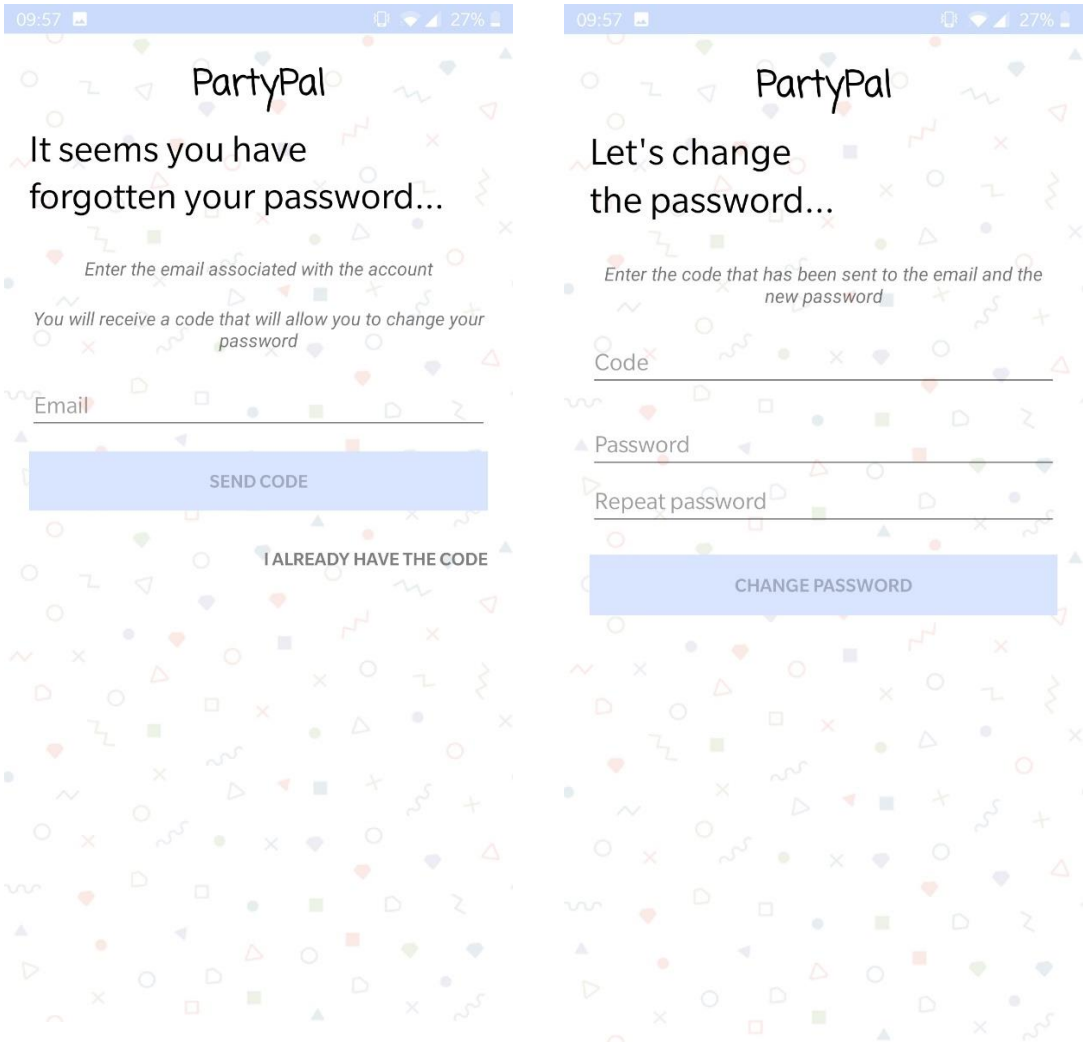
El registre d'usuaris estarà partit en tres vistes per tal de facilitar la introducció de dades i el guiatge de l'usuari per tot el procés. La primera vista demanarà les dades bàsiques: nom, cognom, data d'aniversari, gènere (on les opcions disponibles seran home, dona, altre i "prefereixo no contestar"), mail i contrasenya. Abans de poder passar a la següent vista el sistema comprovarà que el mail no està en ús. En cas que ho estigui, se li indicarà a l'usuari en aquest punt per tal d'evitar que arribi al final del procés de registre i hagi de tirar enrere.

La segona vista permet definir quins són els interessos de l'usuari, és a dir, definir de quin gènere vol que siguin les recomanacions que li mostri l'algoritme de recomanació. Es permet triar només homes, només dones i ambdós gèneres.

Per últim, l'última vista permet triar la imatge principal del perfil. Aquesta imatge es mostrarà a altres usuaris un cop el compte estigui creat.

Tant el gènere de l'usuari, el gènere de les recomanacions, la fotografia del compte i la contrasenya d'aquest es podran canviar posteriorment.

Oblit de contrasenya

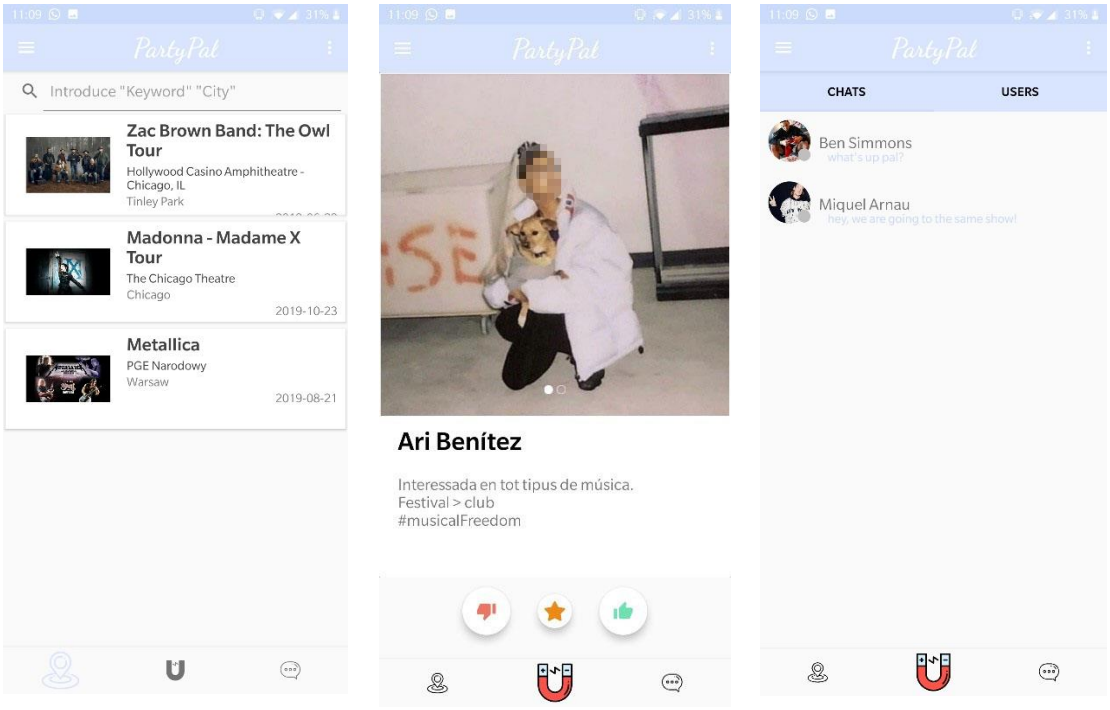


D'esquerra a dreta: mailRecuperació.xml i recuperarContrasenya.xml.

Aquesta funcionalitat permetrà recuperar la contrasenya d'un usuari en cas que l'hagi oblidada.

La primera vista sol·licitarà introduir el correu electrònic que es feia servir per a accedir al compte i es comprovarà si existeix al sistema. En cas que no, se li indicarà a l'usuari. En cas que sí, se li enviarà un correu electrònic amb un codi, que haurà de copiar per tal d'enganxar-lo a la següent vista. En aquesta mateixa vista podrà escriure la nova contrasenya. Si el codi és vàlid i la contrasenya també, s'aplicarà el canvi i es redirigirà a l'usuari a la vista de login. En cas que no ho sigui, se li indicarà a l'usuari.

Funcionalitats principals dins l'app



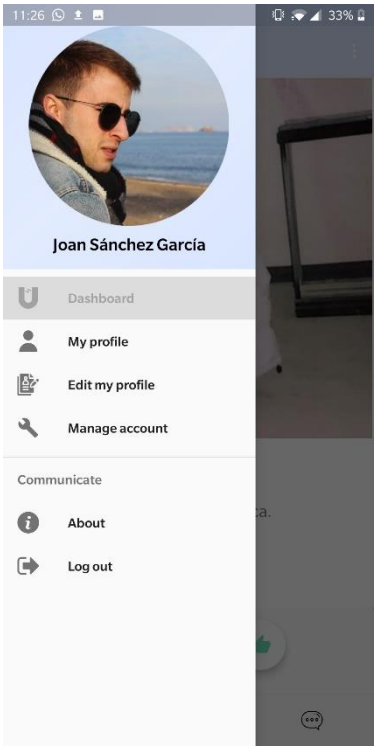
D'esquerra a dreta: eventFinder.xml, matchingScreen.xml, xat.xml.

Un cop s'inicii sessió dins del compte, la primera vista que es carregarà serà la matchingScreen.xml. Aquesta vista comptarà amb dos menús de navegació que estaran presents en totes les finestres de l'aplicació. El primer és el menú lateral, al que es podrà accedir lliscant des de la vora esquerra de la pantalla al centre o clicant les tres línies paral·leles que es troben a la barra blava al costat esquerre del logotipus de l'aplicació. Aquest menú es detalla en el següent apartat. L'altre menú és la barra amb tres opcions que es troba a la vora inferior de la pantalla.

L'opció de més a l'esquerra condueix a la vista eventFinder.xml, la central a matchingScreen.xml i la de més a la dreta a xat.xml.

Des de qualsevol part de l'app es poden clicar aquestes opcions per a ser redirigit a aquestes vistes.

Funcionalitats secundàries dins l'app



A l'esquerra, insideScreen.xml. A sota, d'esquerra a dreta: perfil.xml, editarPerfil.xml i editarCompte.xml.

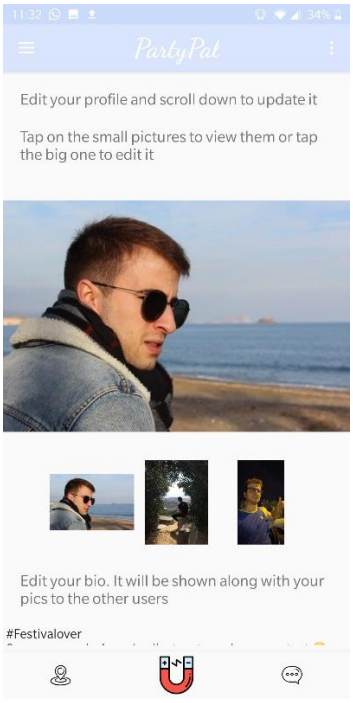
Descrit al punt anterior, al menú lateral es podrà accedir lliscant de la vora esquerra al centre o clicant les tres línies paral·leles de la barra superior i serà accessible en qualsevol vista de l'app un cop iniciada la sessió.

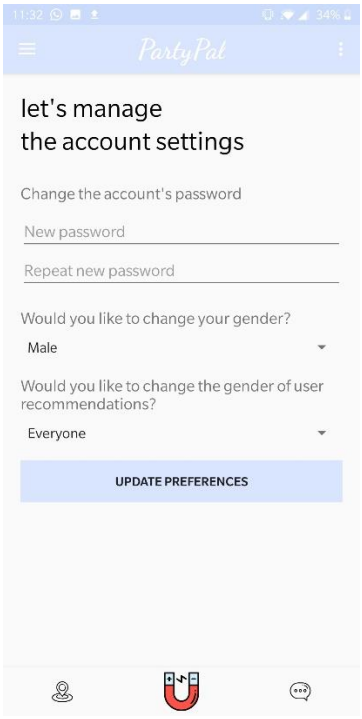
Permetrà accedir a les funcionalitats "veure el meu perfil", "editar el meu perfil" i "editar el compte". A la vegada permetrà accedir també a la informació relativa a l'app i tancar la sessió.

La primera de les funcionalitats, veure el perfil, mostrarà les fotografies que estiguin associades al compte a partir d'una galeria lliscant. Es poden associar d'una a tres fotografies. També mostrarà el nostre nom i la nostra descripció.

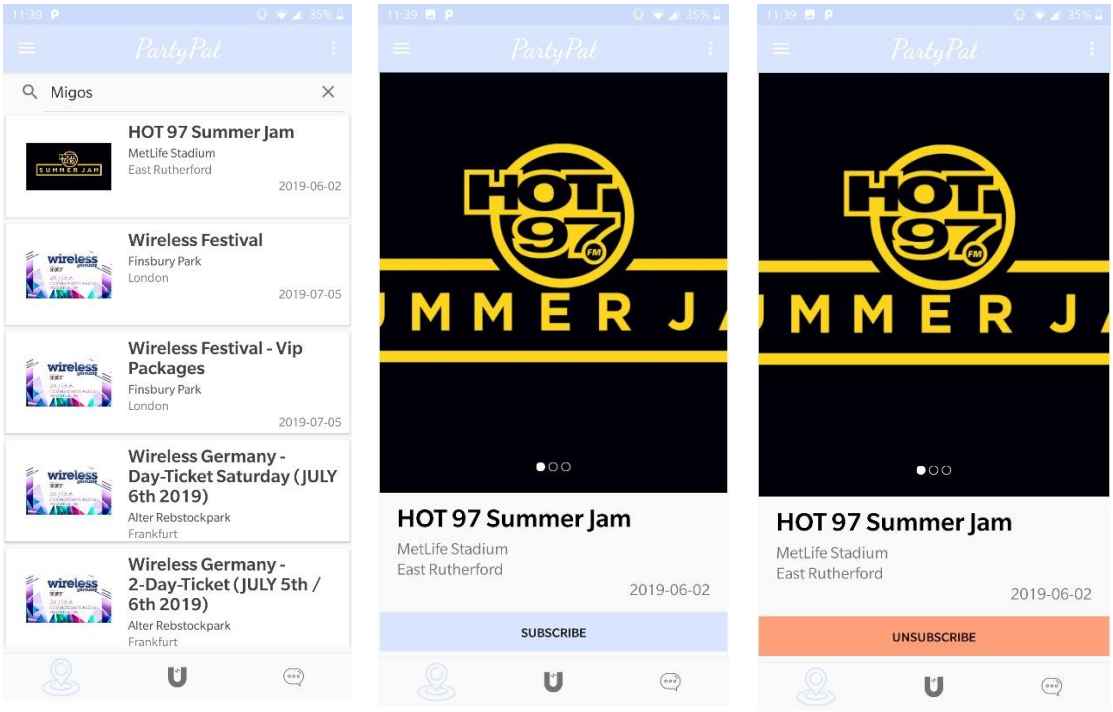
La segona de les funcionalitats ens permet modificar el nostre perfil: permet canviar les fotografies i la descripció, però no el nom de l'usuari.

Per últim, la funcionalitat per editar el nostre compte ens permet canviar la contrasenya el nostre gènere i el gènere de les recomanacions que ens farà l'algoritme de recomanacions.





Cercador esdeveniments



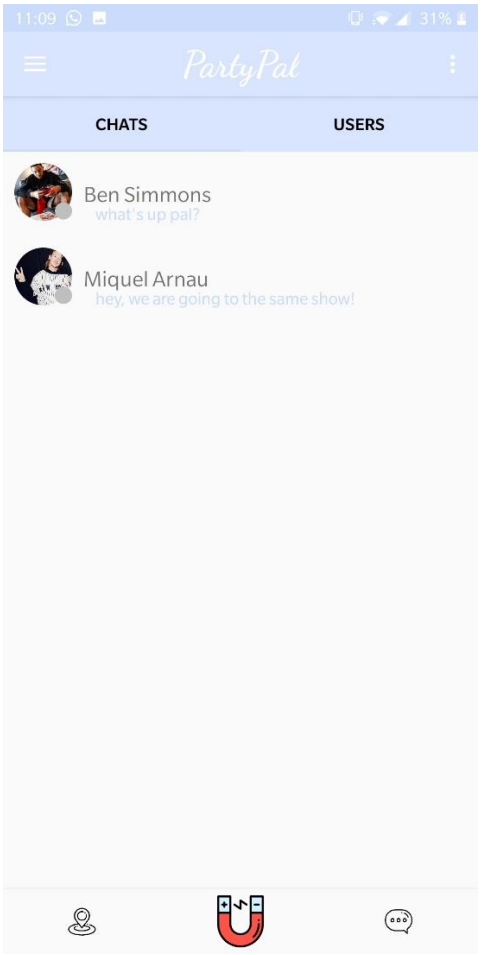
La vista de més a l'esquerra és eventFinder.xml i les altres dos són les dues versions de la vista event.xml.

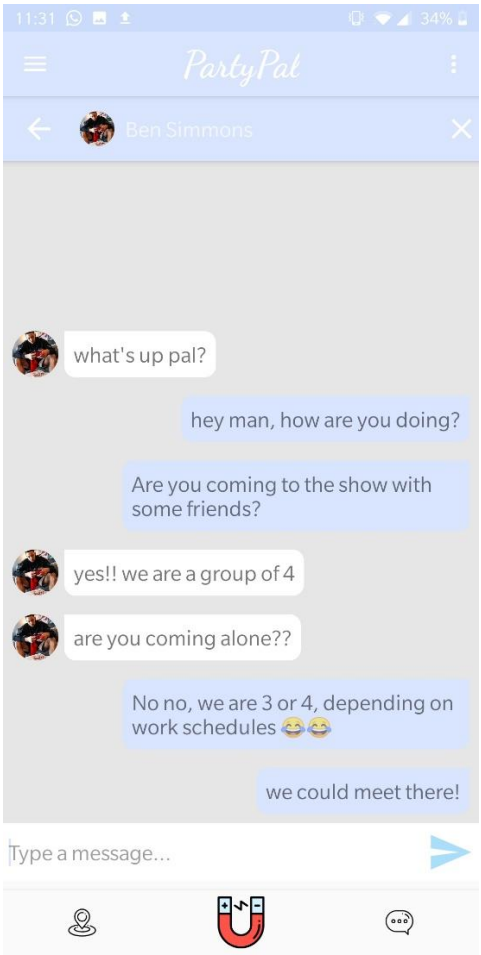
EventFinder.xml contindrà un cercador que permetrà introduir les paraules clau que creiem necessàries per a trobar l'esdeveniment desitjat dins del sistema extern de cercador d'esdeveniments. Aquest cercador espera entre una i dues paraules. En el cas que s'introdueixi només una, interpretarà que estem introduint la paraula clau o *keyword*. En cas que se'n introdueixin dues, la primera serà la *keyword* i la segona la ciutat on volem cercar l'esdeveniment.

Un cop tinguem la llista dels esdeveniments que s'adeqüen a la cerca que hem realitzat, podrem prémer sobre el que més ens interessi per a veure la informació proporcionada. A més, la vista event.xml ens permetrà subscriure'ns o cancel·lar la subscripció en cas que ja l'haguérem realitzat prèviament.

El fet de subscriure's, com ja explicat prèviament, farà que el nostre algoritme de recomanació d'usuaris ens mostri nous perfils que també segueixin aquest esdeveniment. D'igual manera, també apareixerem a l'algoritme de recomanació de més gent.

Sistema de Xat



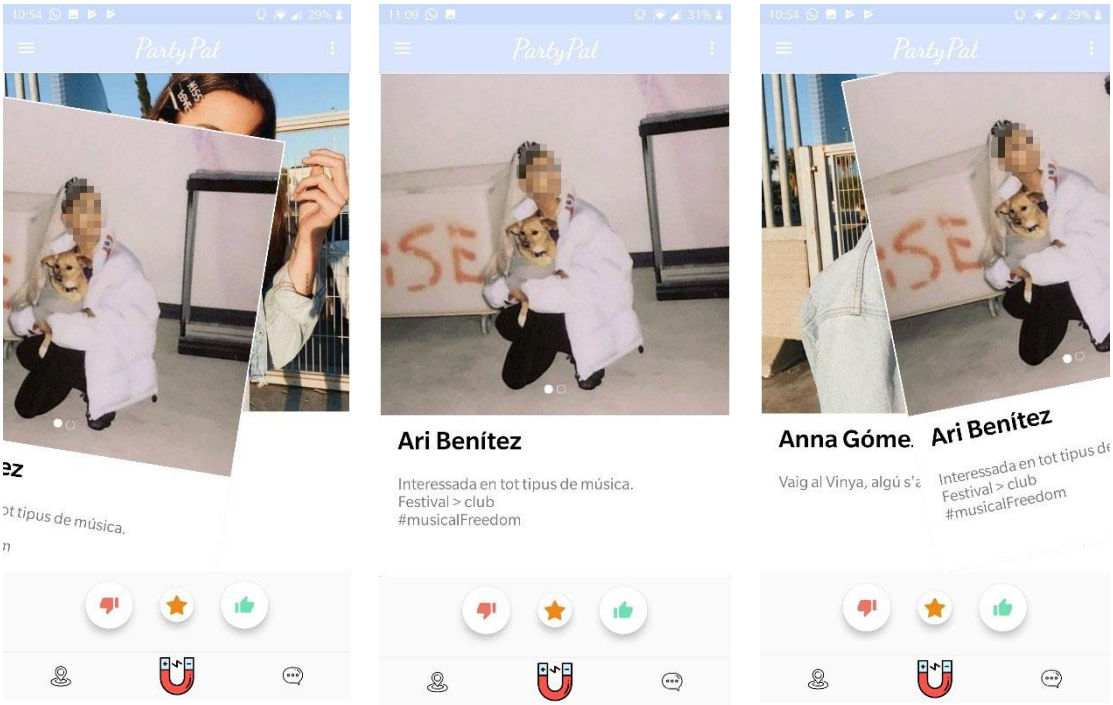


A l'esquerra xat.xml i a la dreta xatUsuari.xml

La llista de xats mostra tots els usuaris amb els quals hem compartit missatges, sense la necessitat obligatòria que siguin matches. Recordem que una de les maneres de fer match era enviant un missatge a l'usuari receptor quan ens apareixia fent servir l'algoritme de recomanació de perfils. Per tant, en aquesta vista tindrem els xats amb els usuaris amb qui som match i amb els usuaris que ens han enviat un missatge de match. Si responem a aquests últims, generarem match. La finestra USERS, que es pot accedir des de xat.xml, llista només els usuaris amb qui tenim match, per tant filtra aquells als que encara no hem respost.

Quan cliquem sobre un xat, sigui match o no, s'obrirà una finestra semblant a xatUsuari.xml on podrem respondre o enviar missatges. Si dins d'aquesta vista premem sobre el nom o la foto de l'usuari se'ns obrirà el seu perfil (vista perfil.xml amb la informació de l'usuari i no la nostra). Si premem sobre la X de més a la dreta, el sistema ens preguntarà si realment volem eliminar aquest xat. Eliminar un xat desfà un match.

Sistema de match



Les tres imatges corresponen a la vista matchingScreen.xml mostrant totes les funcionalitats que es poden fer dins d'aquesta.

Quan obrim l'app o tornem a aquesta finestra se'ns mostrarà una llista en forma de pila de tots els usuaris amb qui el sistema ha considerat que podríem tenir similituds a partir dels esdeveniments que tinguem en comú. La vista inicialment mostrarà un perfil. Dins d'aquest perfil podrem trobar el nom, la descripció i les fotografies de la persona.

Hi haurà dues maneres d'interactuar: la primera serà amb els botons dels dits i l'estrella que es troben sota les targetes dels perfils. El dit cap avall en vermell indica que descartem la recomanació, el dit cap amunt en verd que l'acceptem i a més volem enviar un missatge de match. L'estrella que l'acceptem també, però sense enviar el missatge de match.

Part d'aquesta interacció es podrà fer jugant amb les targetes de forma manual. Si la targeta amb la informació és lliscada cap a l'esquerra, estarem dient que descartem la recomanació. Si la llisquem cap a la dreta, que l'estem acceptant i a més volem enviar missatge de match. La funcionalitat d'acceptar sense enviar missatge només es podrà realitzar prement l'estrella dels botons.

Acceptem o rebutgem una recomanació, el sistema ja tindrà preparada la següent. Això seguirà passant fins que ja no en quedin més.

7.2.2. Mapa navegacional de la capa de presentació

La Figura 10 mostra les navegacions realitzables entre les vistes del sistema. El nom de cada vista és l'especificat en l'apartat anterior. Els comentaris de cada fletxa es refereixen a "Botó o enllaç clicat [aclariments]".

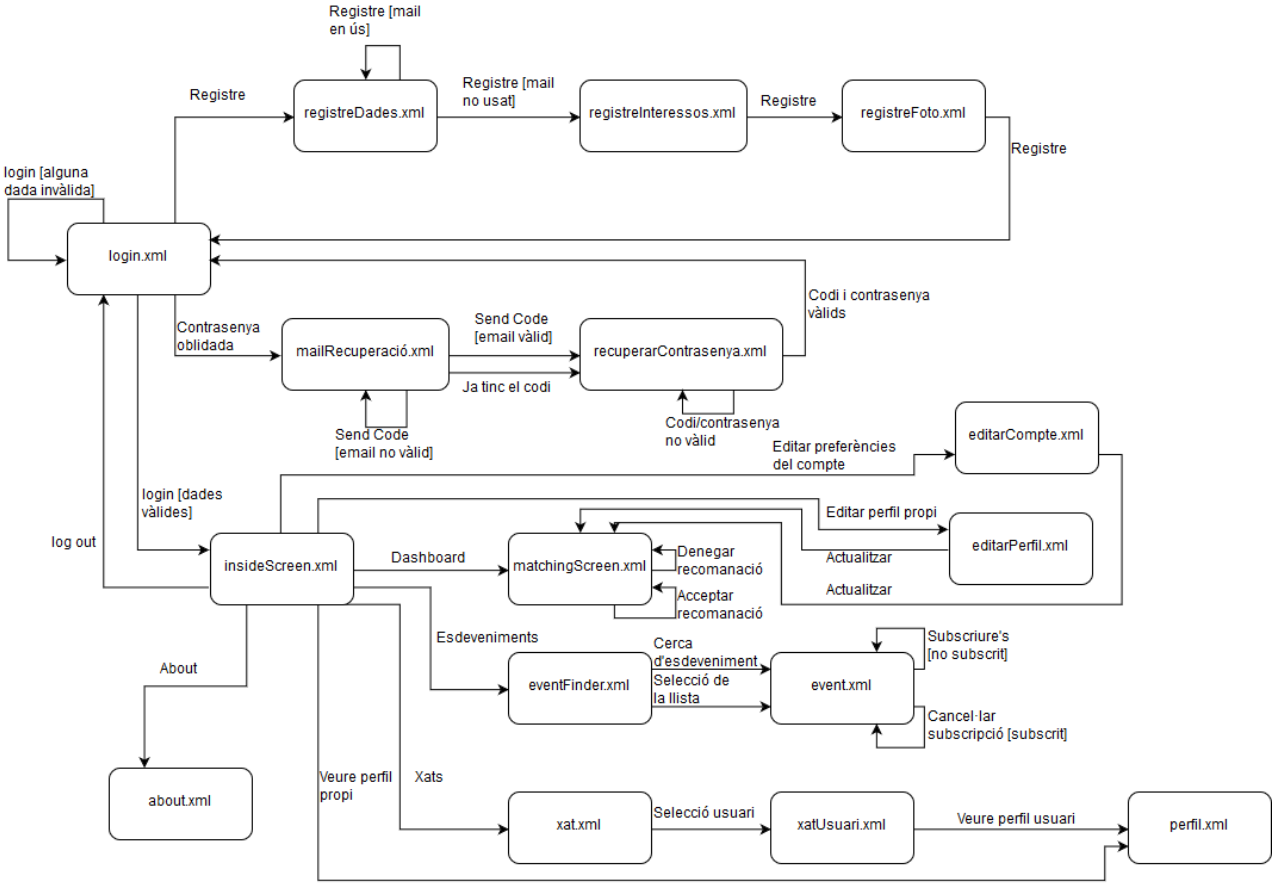


Figura 10: mapa navegacional de les vistes del sistema

7.2.3. Disseny intern de la capa de presentació

La creació d'una interfície d'usuari és una tasca complexa donada la quantitat de classes, interfícies i adaptadors necessaris per a adaptar els requeriments a la tecnologia sobre la qual es construirà. És per això que el resultat final, encara que estigui compost per poques vistes, pot necessitar un gran nombre de classes per a gestionar-les.

És per aquest motiu que no s'ha cregut oportú adjuntar tot el diagrama de classes intern que conforma l'app mòbil atès el poc ús pràctic. Es mostra, això sí, l'explicació del disseny amb una simplificació d'aquest.

La figura 11 mostra una abstracció de les classes que han intervingut en la gestió del registre o alta d'usuaris. Cada classe està posicionada en la capa a la qual pertany.

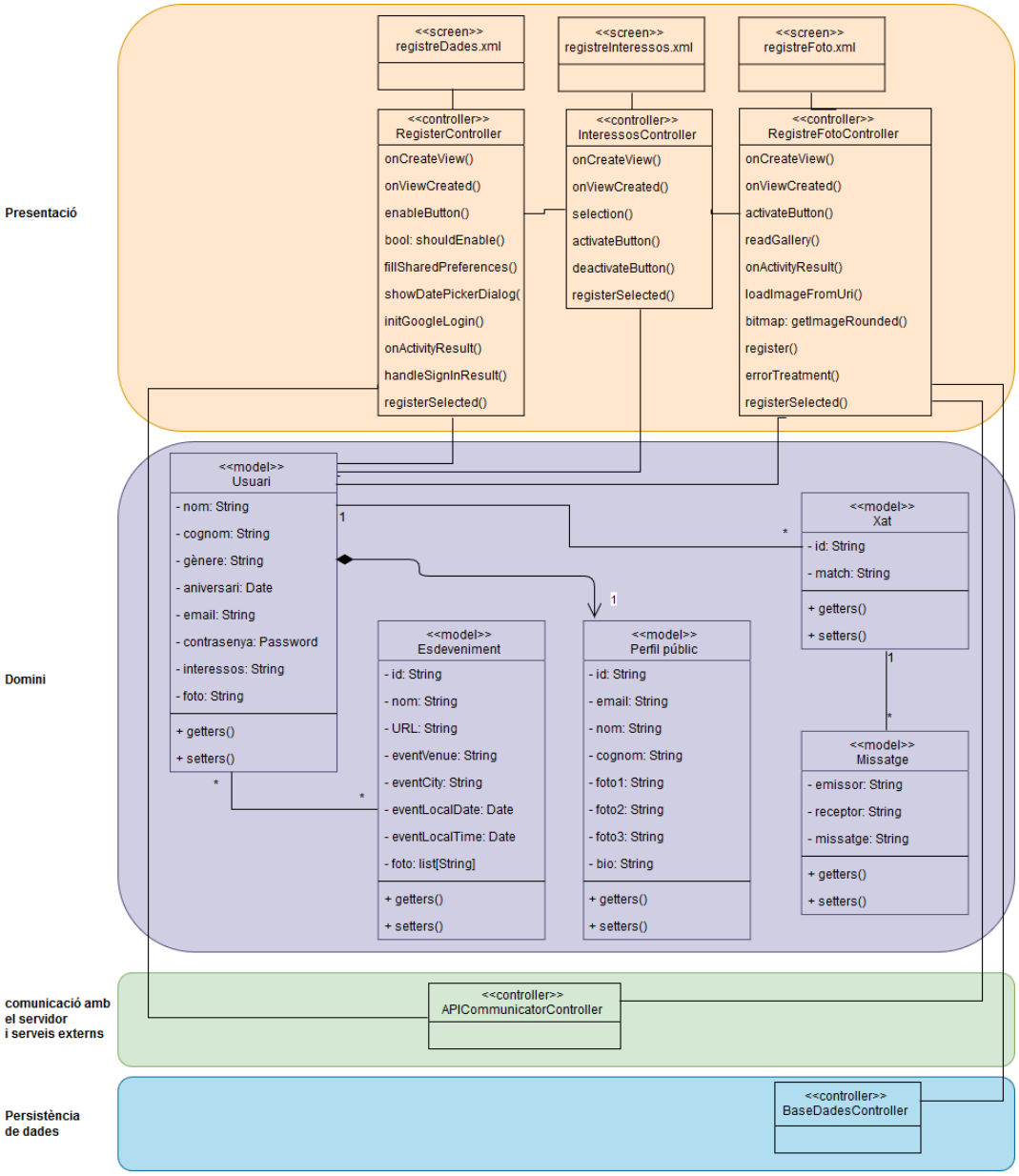


Figura 11: Diagrama de classes relatiu a la gestió d'alta d'usuari en el front end

Tal com es pot apreciar a la figura 12, i ja s'ha explicat anteriorment, el registre d'usuari constarà de 3 vistes, cadascuna recull una informació diferent. Cada vista serà gestionada pel seu controlador, que anirà omplint una instància del model "Usuari". Aquesta instància anirà passant de controlador en controlador fins a arribar al RegistreFotoController. En aquesta vista l'usuari podrà seleccionar la imatge que vol que estigui associada al seu compte, que s'haurà de guardar en memòria local fins que es comprimeixi i s'enviï al servidor. Un cop en aquest punt, el controlador enviarà la petició i una representació de l'objecte al servidor a través de l'APICommunicatorController. Depenent de la resposta i el temps que trigui en arribar, es farà un tractament o un altre: en cas que sigui una resposta positiva, s'entendrà que l'usuari ha estat registrat amb èxit i es conduirà l'usuari a la vista de login. En cas que sigui negativa, s'informarà a l'usuari del motiu.

Per a facilitar la comprensió d'aquest recorregut i mostrar com interaccionen els diferents components que conformen la funcionalitat, s'adjunten les figures 12 i 13, que mostren el diagrama de seqüència per al registre d'un usuari. El diagrama mostra gràficament les interaccions que l'usuari realitza amb el sistema i com reacciona aquest.

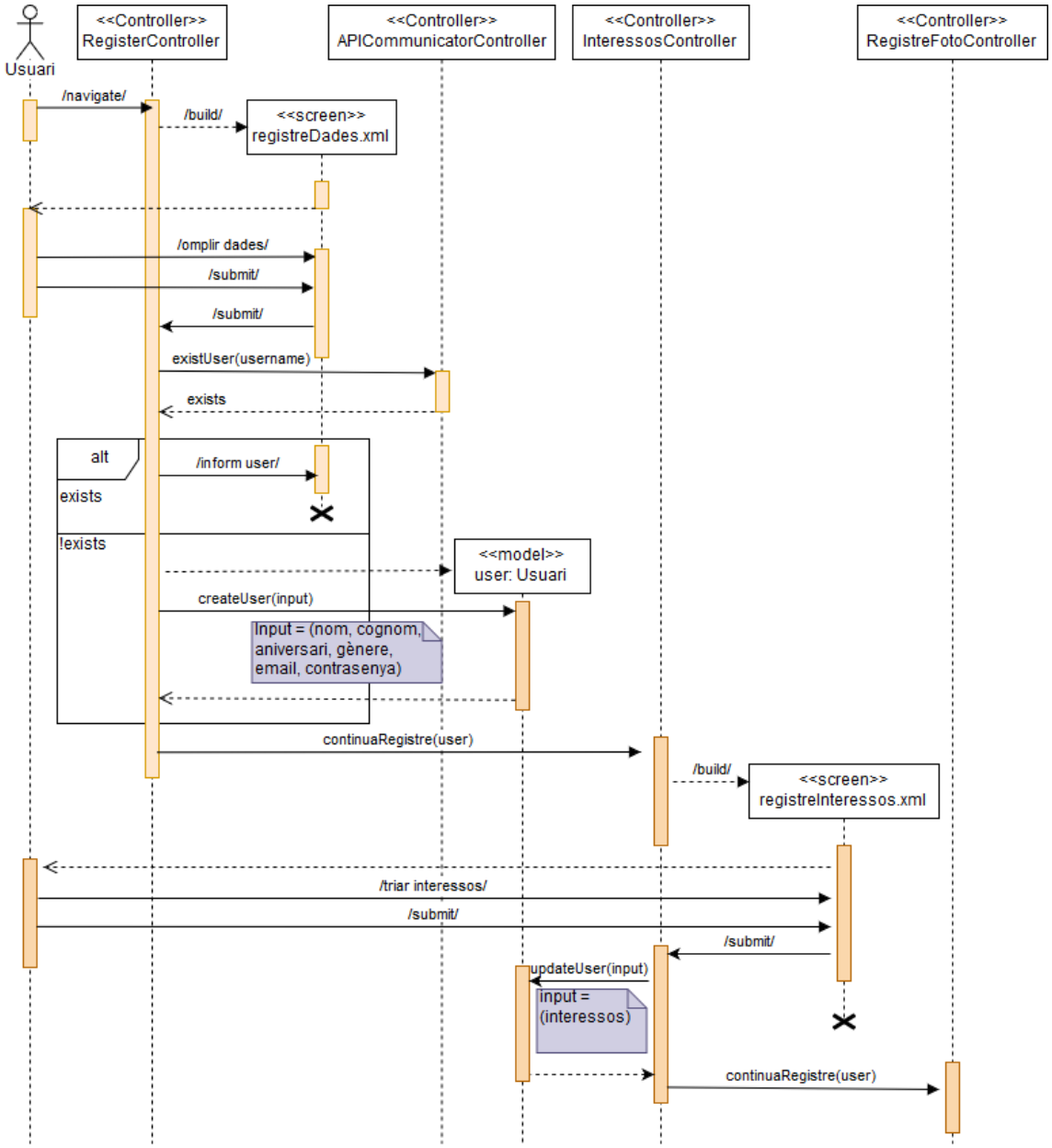


Figura 12: Diagrama de seqüència relatiu a la gestió d'alta d'usuari al component front end. Primera part

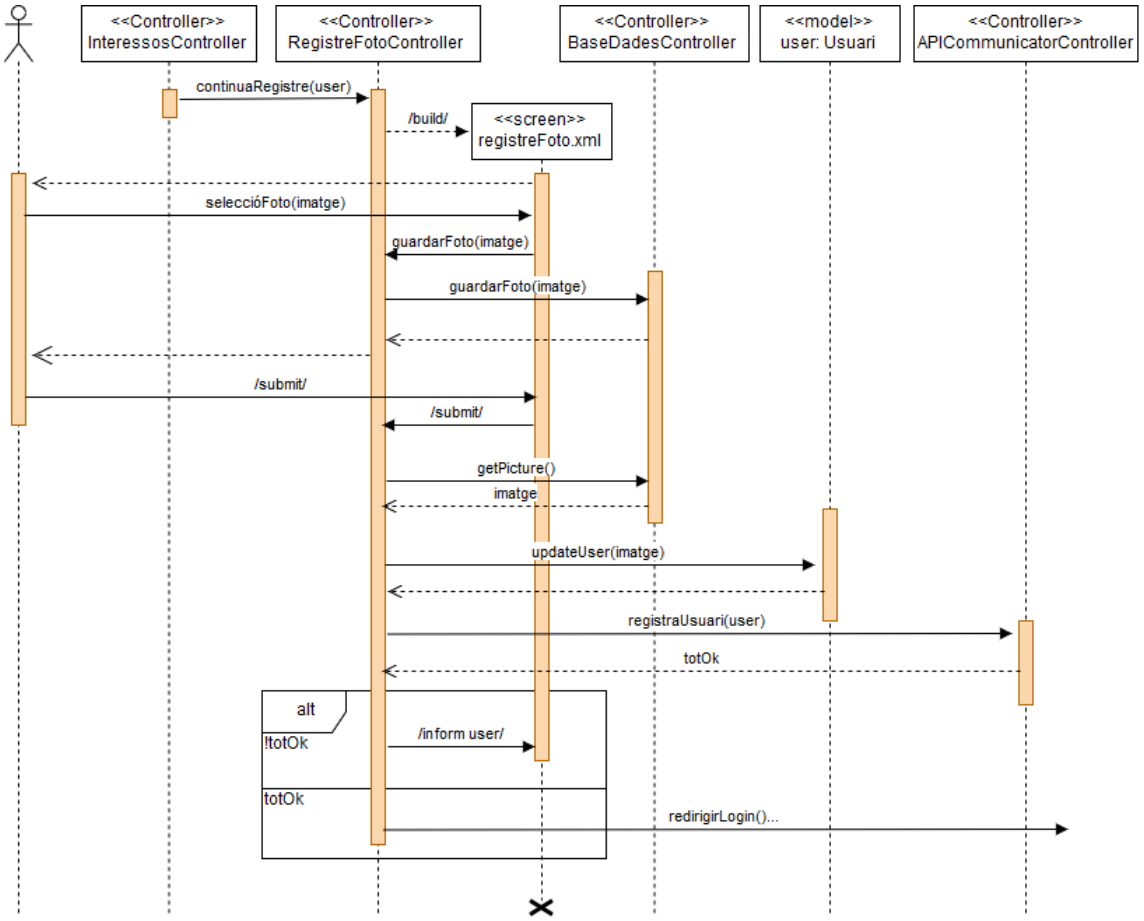


Figura 13: Diagrama de seqüència relatiu a la gestió d'alta d'usuari al component front end. Segona part

Detalls més específics sobre la implementació de l'app seran proporcionats en el capítol següent.

7.3. Disseny del servidor (back end)

Un cop descrit el disseny de l'app mòbil, es procedeix a descriure l'arquitectura del servidor back end. Atès que el servidor funcionarà com una API, a continuació es llisten les crides que haurà d'acceptar. Més endavant es mostra un diagrama de classes i la continuació de la funcionalitat de registre en la banda del servidor.

7.3.1. Mètodes oferts

Com s'ha explicat a l'apartat d'arquitectura lògica, les crides seran agrupades en paquets depenent de l'entitat sobre la qual actuen. S'han definit 4 paquets: usuari, esdeveniment, match i autenticació de compte.

Els mètodes permesos són GET per a llegir, POST per a crear, PUT per a editar i DELETE per a eliminar.

Usuaris: Gestió bàsica

POST /users

Registre d'usuaris. La informació relativa a l'usuari s'haurà d'enviar al cos de la crida.

GET /users/{email}

Retorna un usuari en concret identificat pel seu mail.

GET /users/{email}/exists

Indica si un determinat usuari en concret identificat pel seu mail existeix o no.

DELETE /users/{email}/delete

Elimina del sistema un usuari en concret identificat pel seu mail.

Usuaris: Recuperació de contrasenya

POST /recover

Activa el procés de recuperació de contrasenya pel compte associat al mail inclòs al cos de la crida.

PUT /recover/{token}

Si el token indicat a la crida coincideix amb l'enviat, es canvia la contrasenya inclosa al cos de la crida.

Usuaris: Gestió perfil d'usuari

GET /profile/{email}

Retorna el perfil de l'usuari associat al mail.

PUT /profile

Actualitza el perfil de l'usuari indicat al cos de la crida amb la informació també aquí inclosa.

Usuaris: Gestió del compte

PUT /config

Actualitza la configuració del compte de l'usuari indicat al cos de la crida amb la informació també aquí inclosa.

GET /config/{email}

Retorna la configuració del compte del mail sol·licitat.

Match

GET /match/recommendations/{email}

Retorna un conjunt dels perfils que el sistema recomana a l'usuari associat amb el mail, si n'hi ha cap.

POST /match/like

L'usuari u1 indica que li agrada la recomanació del perfil de l'usuari u2. Els mails dels dos usuaris estan definits al cos de la crida.

POST /match/dislike

L'usuari u1 indica que no li agrada la recomanació del perfil de l'usuari u2. Els mails dels dos usuaris estan definits al cos de la crida.

POST /match/save

L'usuari u1 indica que el sistema guardi la recomanació del perfil de l'usuari u2. Els mails dels dos usuaris estan definits al cos de la crida.

DELETE /match/unmatch

L'usuari u1 indica que vol desfer el match amb l'usuari u2.

Esdeveniment

GET /event

Retorna els esdeveniments que coincideixin amb la paraula clau i la ciutat definides al cos de la crida. Aquesta crida cerca al repositori d'esdeveniments del servei extern.

GET /event/{eventid}

Retorna l'esdeveniment eventid en particular. Aquesta crida cerca al repositori d'esdeveniments del servei extern.

POST /event/subscribe

Crea la subscripció de l'usuari a l'esdeveniment en concret. Tant el mail de l'usuari com l'id de esdeveniment van definits al cos de la crida.

PUT /event/unsubscribe

Cancel·la la subscripció de l'usuari a l'esdeveniment en concret. Tant el mail de l'usuari com l'id de esdeveniment van definits al cos de la crida.

GET /events/{userid}

Retorna un llistat d'esdeveniments als quals l'usuari definit s'ha subscrit.

GET /events/exists

Indica si la subscripció entre l'usuari i l'esdeveniment definits al cos de la crida existeix al sistema.

Autenticació

PUT /login

Es prova d'iniciar sessió amb l'usuari i la contrasenya definits al cos de la crida.

7.3.2. Diagrama de classes del domini

Atès el volum de classes per definir en el domini i la quantitat de propietats internes i crides que tenen, realitzar un diagrama únic mostrant-ho tot era poc pràctic per a la resolució gràfica màxima que un document escrit pot aportar. Per a resoldre aquest problema, les figures 14, 15 i 16 mostren els diagrames de classes de cada paquet, incloent-hi atributs, mètodes, dependències entre classes i les excepcions que es faran servir.

Per a crear la imatge global del sistema, la figura 17 mostra el diagrama de classes complet, però sense els mètodes i atributs interns, només amb el nom de la classe. Per a augmentar la visibilitat, també s'han eliminat les classes de les excepcions, que només són cridades pel paquet on es troben definides.

El paquet "auth" referent al sistema d'autenticació serà explicat a l'apartat d'implementació. En aquest apartat de disseny només ens cal saber que el paquet necessita crear una instància d'un usuari i que s'ocupa, a partir de les credencials que rebi, de provar d'iniciar la sessió del compte.

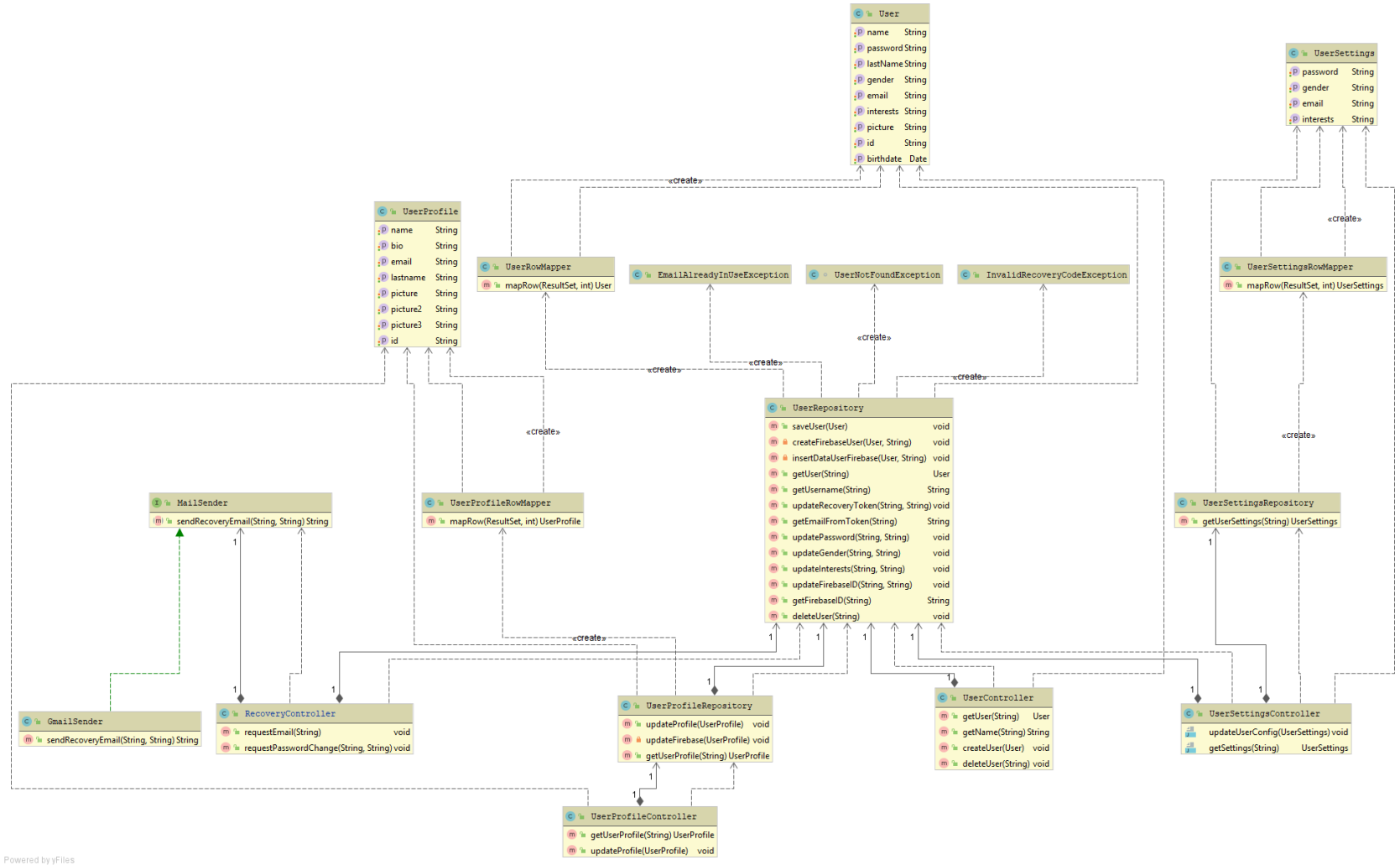


Figura 14: Diagrama de classes per al paquet d'usuari

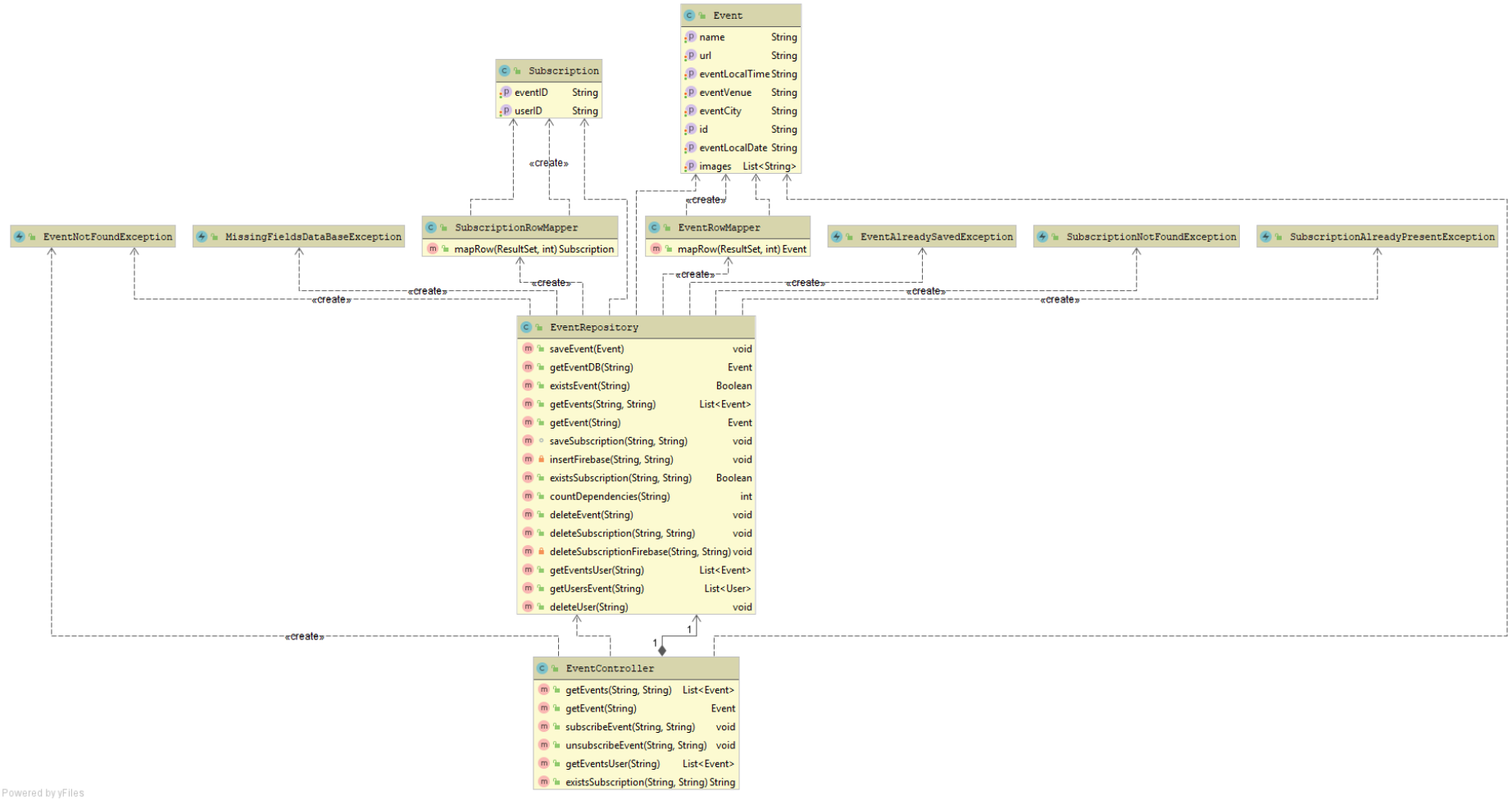


Figura 15: Diagrama de classes per al paquet d'esdeveniment

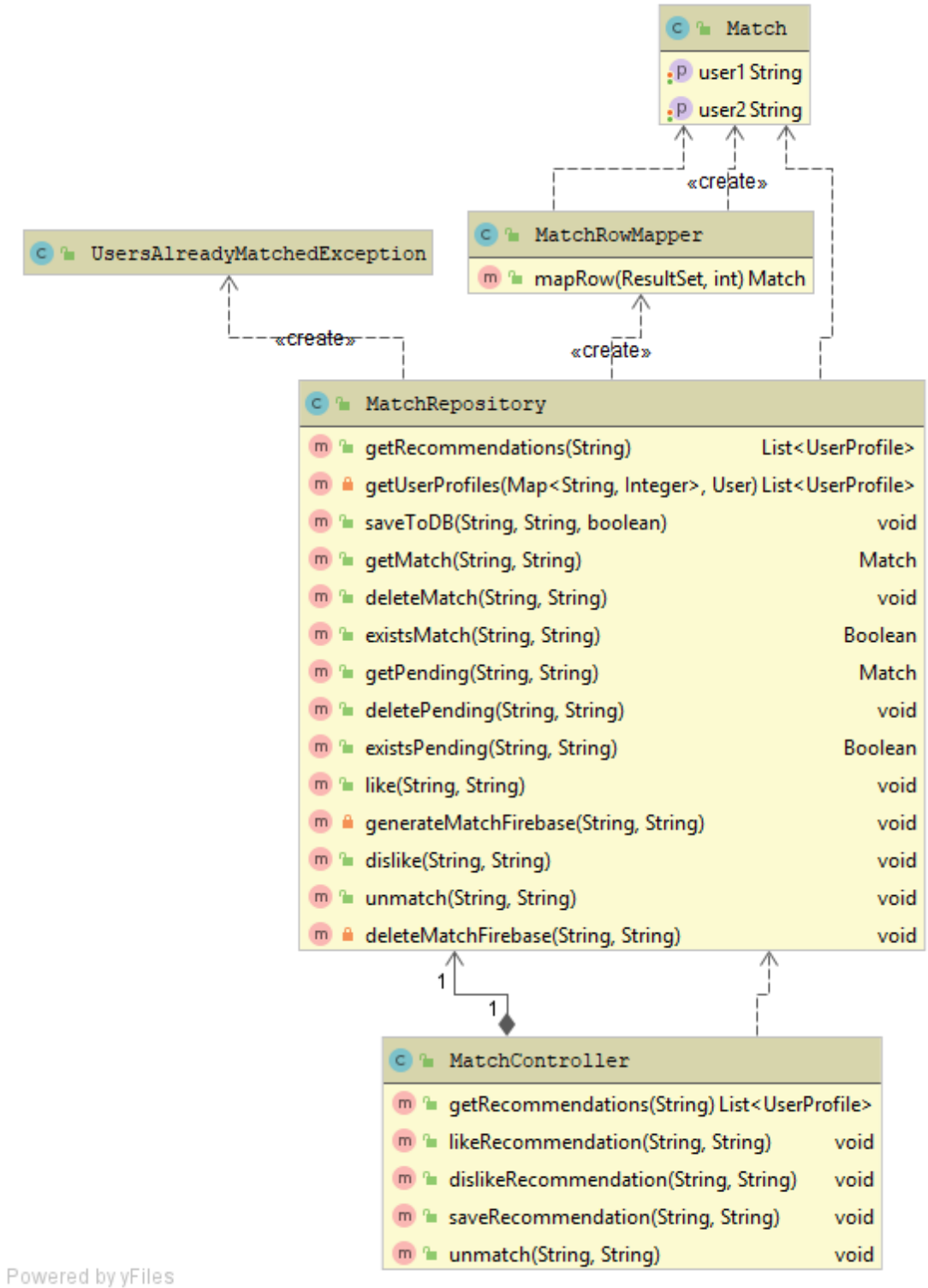


Figura 16: Diagrama de classes per al paquet de match

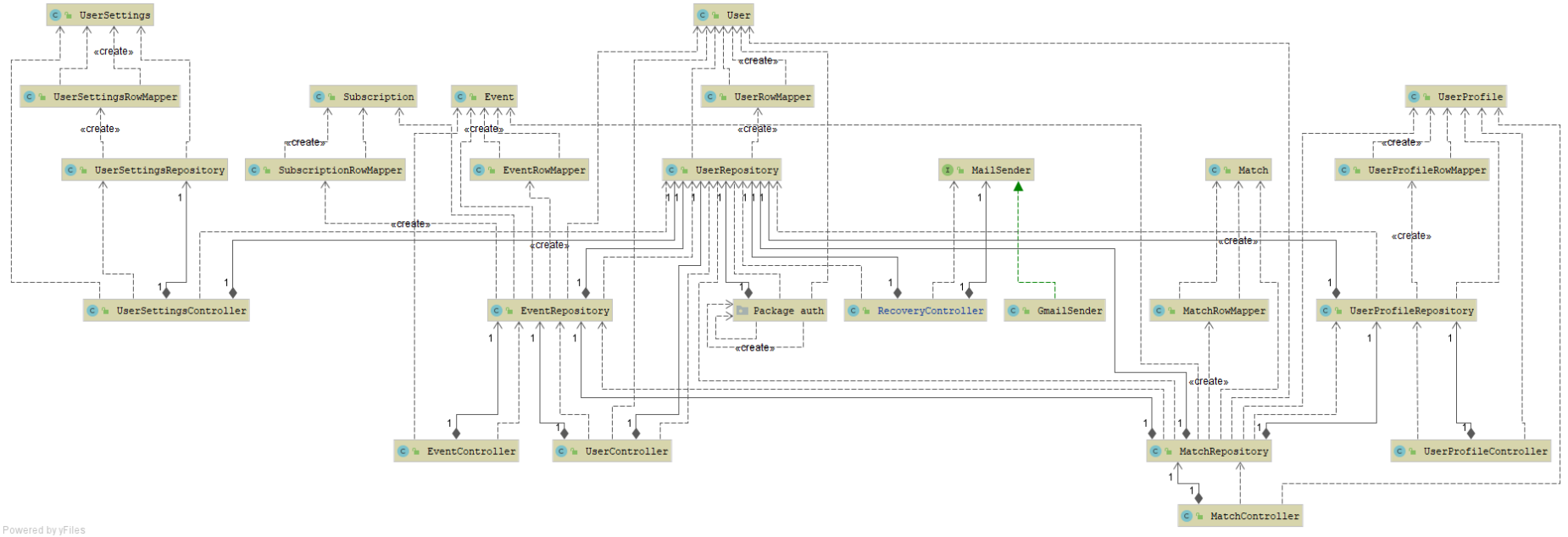


Figura 17: Diagrama de classes de l'aplicació

7.3.3. Diagrama de seqüència

Seguint amb l'exemple del registre començat a l'apartat del disseny de l'app mòbil, la figura 18 mostra el recorregut que faria la crida de l'alta d'usuari des que arriba al servidor. Només arribar, serà interceptada per un *handler* que farà de front controller (més detalls a l'apartat següent). Aquest controlador llegirà a quina URI (Identificador de Recursos Uniforme, cadena de caràcters única dins el sistema per a identificar recursos) anava dirigida la crida i l'enviarà al controlador corresponent, en aquest cas, l'UserController.

Un cop arribi la crida al seu controlador, es convertirà el cos d'aquesta en un objecte del sistema per a facilitar el trànsit entre classes. El propi model d'usuari tindrà la funcionalitat per a transformar el cos de la crida en una instància d'ell mateix. El controlador passarà aquesta instància al seu repositori, que s'encarregarà de guardar-la a la base de dades. Per a fer això serà necessari fer servir la interfície del gestor de la base de dades que serà una *template* de la seva implementació (més detalls a l'apartat següent). Si l'acció de guardat és correcta, es retornarà un OK a l'app mòbil. Si fos errònia, es llançarà una excepció que retornarà el motiu de l'error, en aquest cas, que el sistema ja té un usuari fent servir aquest mail com a clau primària.

En el cas que es volgués llegir de la base de dades, caldria fer servir el RowMapper del model d'usuari perquè la *template* del gestor de la base de dades sabès transformar del format en què estiguin guardades a un objecte del sistema.

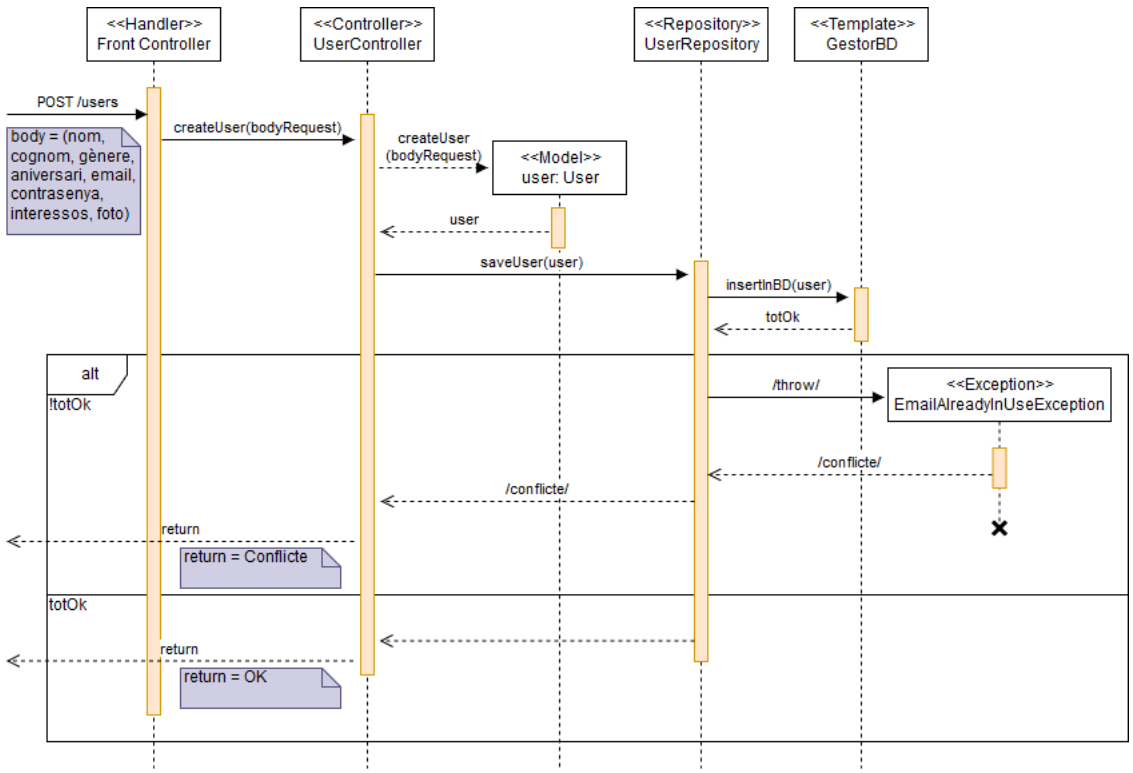


Figura 18: Diagrama de seqüència relatiu a la gestió d'alta d'usuari al servidor

7.3.4. Patrons de disseny

Diversos patrons de disseny han estat aplicats en el disseny de l'arquitectura del servidor. Hi ha alguns que són més específics de la implementació triada i s'explicaran més endavant. Tots els patrons que s'anomenen a continuació han estat extrets de *Design Patterns: Elements of Reusable Object-Oriented Software* [33]

Patrons de disseny per a la creació

- Builder [34]: el patró Builder permet construir un objecte pas a pas i no tot a la vegada. Es basa en el concepte de crear una classe estàtica que actuarà entre el constructor real i les peticions del client i funcionarà com a espai d'emmagatzematge temporal. El client anirà enviant les dades de l'objecte que vol construir a aquesta classe i quan ja ho tingui tot cridarà al mètode build(), que finalment construirà l'objecte.
  Encara que no s'ha creat la classe estàtica intermèdia per a cada model de dades del sistema, els constructors d'aquests requereixen el mínim d'atributs per a poder crear-se, permetent afegir els altres quan calgui a partir dels setters i getters.

Patrons de disseny estructural

- Adapter [35]: tota la gestió del servei extern per al cercador d'esdeveniments es fa a partir del patró adaptador. El patró adaptador crea una interfície que permet al sistema usar la interfície d'una entitat ja creada, és a dir, actua com a pont entre el sistema i el servei. En el nostre cas, va caldre crear els mètodes per a comunicar el sistema amb la interfície del cercador d'esdeveniments.
- Proxy [36]: tot i que aquesta part de l'explicació es troba a la frontera entre disseny i implementació, donat que s'ha fet referència al *handler* al diagrama de seqüència s'explicarà aquí. Tal com indica Pankaj [36], "el patró Proxy proporciona un substitut o un mètode per a controlar l'accés a un objecte del sistema". El *handler* és la porta d'entrada al sistema des de l'exterior: qualsevol crida passarà per ell abans de continuar. El *handler* redirigirà la crida al controlador que en realitza el tractament a partir de la seva URI o retornarà un error en cas que aquesta no la tracti cap controlador, és a dir, que el mètode no és suportat pel sistema.

Patrons de disseny per al comportament

- Template [37]: Igual que amb el patró Proxy, el Template també es troba a la frontera entre disseny i implementació, però de nou, per al·lusions en apartats anteriors s'explicarà aquí. Tal com indica Pankaj [37], "El patró Template defineix els passos per executar un algorisme i pot proporcionar una implementació per defecte que pot ser comuna per a totes o algunes de les subclasses.". D'aquesta manera, independentment de la implementació que triem per a la base de dades, el patró s'ocuparà de gestionar l'algoritme de lectura i escriptura per nosaltres, facilitant la feina del programador i minimitzant l'acoblament entre el sistema i la base de dades. Més detalls d'aquest patró seran proporcionats al capítol d'implementació.
- Chain of Responsibility [38]: "El patró chain of responsability s'utilitza per aconseguir un acoblament baix en el disseny de programari on es transmet una petició del client a una cadena d'objectes per processar-los. En cada pas és l'objecte de la cadena qui decidirà si processa ell mateix la petició o la passa al següent.". Per a cada petició el recorregut màxim serà el mateix: Front Controller, el controlador del model, el repositori del model i accés a la base de dades amb implicació del Row Mapper del model.

7.4. Esquema de la base de dades

Per a acabar amb aquest capítol es mostra l'esquema de la base de dades. La taula "User" engloba tota la informació que es pot aconseguir de l'usuari fent ús de totes les funcionalitats implementades, encara que en alguns casos hi haurà valors nuls si l'usuari decideix no usar-les. La taula "Event" recull la informació dels esdeveniments als quals algun usuari s'ha subscrit. Atès que fer la crida al servei extern és una operació costosa i en alguns casos fins i tot limitada (algunes APIs estableixen un límit de peticions per client), quan un esdeveniment en concret registri una subscripció, és a dir, es creï una fila a la taula "Subscriptions", les dades de l'esdeveniment es guardaran en memòria. D'igual manera, si quan els usuaris cancel·len la subscripció ja cap fila de "Subscriptions" depèn de l'esdeveniment, aquest s'eliminarà de la base de dades. D'aquesta manera fem que el mètode per a carregar els esdeveniments dels usuaris sigui més ràpid que realitzar una cerca al servei extern, on no som responsables de la velocitat de resposta.

Les altres dues taules, "Match" i "Pending" serveixen per a gestionar l'algoritme de recomanació. Quan un usuari doni like o guardi un altre usuari, es registrarà a la taula "Pending" (en aquest cas l'ordre de factors sí que altera el producte: a1,a2 vol dir que a l'usuari a1 li ha agradat a2). Si l'altre usuari també dona like, s'eliminarà la fila d'aquesta taula i s'afegirà a la de "Match" (en aquest cas, l'ordre dels factors és indiferent). Si dona que no, s'esborra la fila de la taula "Pending".

User (email, ID, name, lastName, birthdate, gender, password, interests, recovery, Picture, picture2, picture3, bio)

Check {ID} is NOT NULL

Check {name} is NOT NULL

Check {lastName} is NOT NULL

Check {birthdate} is NOT NULL

Check {gender} is NOT NULL

Check {password} is NOT NULL

Check {interests} is NOT NULL

Event (ID, name, url, eventVenue, eventCity, eventLocalDate, eventLocalTime, picture1, picture2, picture3)

Check {name} is NOT NULL

Check {url} is NOT NULL

Check {eventCity} is NOT NULL

Check {eventLocalDate} is NOT NULL

Subscriptions(userID, eventID)

{userID} referencia User

{eventID} referencia Event

Match (user1, user2)

{user1} referencia User

{user2} referencia User

Pending (user1, user2)

{user1} referencia User

{user2} referencia User

8. Implementació

Un cop es tenen clar els requisits del sistema i el disseny de l'arquitectura que els satisfarà, és hora de dur-los a la pràctica. Aquest capítol detalla els aspectes tècnics del procés d'implementació: metodologies seguides, eines fetes servir, enfocaments de programació i determinats patrons que han sorgit d'usar les tecnologies aplicades finalment. Com al capítol anterior, es partirà l'explicació en els dos components que formen el sistema: l'aplicació mòbil front end i el servidor back end. A més, també es descriurà el procés d'implementació del sistema d'integració continua que s'ha fet servir.

Com ja s'ha explicat al capítol de metodologia, aquest projecte n'ha seguit una inspirada en *Scrum,* adaptant-se a una estratègia de desenvolupament incremental i possibilitant un encavalcament entre les diferents fases del projecte. El temps disponible per a programar el sistema es va partir en *sprints* de dues setmanes de duració i es van repartir els casos d'ús identificats entre aquests *sprints*. D'aquesta manera era més fàcil pel programador determinar quina feina havia de tenir feta pel final de la iteració i organitzar-se com volgués dins d'ella. Per aquest motiu, algunes funcionalitats s'han desenvolupat de forma atòmica i altres s'han anat programant en paral·lel.

8.1. Implementació de la app mòbil (front end)

8.1.1. Definició de versions de codi

Tot i que el setembre de 2018 es va alliberar la versió 11 de Java, aquest projecte s'ha construït amb Java 8 atès l'extensa comunitat que encara dona suport i la fiabilitat que aquesta versió ha construït durant els anys en què ha estat la referència d'aquest llenguatge de programació.

Pel que fa a Android, es compila l'app amb el nivell d'SDK 28, que equival a Android 9. Android permet definir un rang de versions de compatibilitat: com a versió màxima s'ha triat la 28 i com a mínima la 23, que equival a Android 6.0. D'aquesta manera, no només es compleix amb el requisit no funcional "el sistema ha de funcionar en la majoria de dispositius Android", ja que la versió 23 és present al 71% dels dispositius, sinó que es garanteix que es podrà treballar amb el paradigma de disseny del *Material Design* proposat per *Google* i gestionar els permisos en temps d'execució [39].

8.1.2. Arquitectura de l'app mòbil (front end)

Per la gran quantitat d'avantatges que aporta, de cara al desenvolupament de la interfície d'usuari s'ha fet servir la filosofia dels fragments. Tal com defineix la mateixa guia de desenvolupador de *Google* [40], "Un Fragment representa un comportament o una part de la interfície d'usuari en una Activity. Pots combinar múltiples fragments en una sola activitat per crear una IU multi panell i tornar a fer servir un fragment en múltiples activitats. Pots pensar en un fragment com una secció modular d'una activitat que té el seu cicle de vida propi, rep els seus propis esdeveniments d'entrada i que pots afegir o treure mentre l'activitat s'estigui executant (una mena de "sub activitat" que pots tornar a utilitzar en diferents activitats)."

De forma gràfica, es representa la definició de Fragment en la Figura 19:

Figura 19: Dos fragments, mostrats en configuracions diferents per a la mateixa activitat en diferents mides de pantalla. Font: desarrollador-android.com

Una metàfora per entendre com funcionen els fragments i les activities seria la dels projectors antics o projector d'acetats (Figura 20). Les activities serien el projector, oferint la tecnologia necessària per a poder projectar les diapositives. Imaginem que cada projector només pogués projectar una diapositiva perquè aquesta estigués enganxada al vidre reflectant. La transacció de diapositiva a diapositiva seria lenta, ja que caldria apagar el projector, moure'l per col·locar el següent, encendre'l i enquadrar-lo perquè la imatge es veiés bé. Igual passa si es realitza tota l'aplicació amb activities: per a cada transició, l'activity actual hauria de ser destruïda o aturada, introduïda a la pila d'activitats i llavors construir i iniciar la nova.

Figura 20: projector d'acetats. Font: mercadolibre.com.mx

Aquest problema se soluciona si el projector no s'ha de moure i l'únic que varia són les diapositives que es posen a sobre. D'aquesta manera la presentació és molt més fluida, atès que el temps necessari per a canviar de vista és molt inferior a l'anterior aproximació. Un altre avantatge és que les diapositives les pot fer servir qualsevol projector, ja que no estan associades a un en particular, ajudant a la seva reutilització. Igual passa amb els fragments: aquests són vistes lligades a la seva lògica de gestió, però poden ser usats per qualsevol activity. A més, la gestió de la vida d'un fragment és molt més fàcil que la d'una activity: es redueix a saber tractar la *BackStack*. Aquesta és la pila de tots els fragments que s'han fet servir des que s'ha iniciat l'activity. D'aquesta manera, cada cop que es passa d'un fragment a un altre, el primer es fica a la pila i cada cop que es premi el botó enrere, es recupera el fragment que estava a la cima fins que no en quedi cap.

Per tant, es faran servir els fragments com les vistes de l'aplicació i les activities com contenidors dels fragments.

En l'app s'han fet servir dues activities: MainActivity i InsideActivity. La primera s'ha encarregat de la gestió de tots els fragments i la seva lògica corresponent de les vistes de fora l'aplicació, això inclou el login, el registre i la recuperació de contrasenya, és a dir, totes les funcionalitats que es poden fer sense requerir tenir un compte d'usuari actiu. L'InsideActivity s'encarrega de la gestió de tot l'interior de l'aplicació.

El primer que fa l'InsideActivity és sol·licitar al servidor tota la informació bàsica de l'usuari necessària per a inflar correctament les vistes. Aquesta informació inclou el nom i la fotografia que es mostraran al menú lateral, el gènere i els seus interessos. Aquesta informació es guarda a la memòria *SharedPreferences,* que és una petita base de dades local al dispositiu només accessible per l'aplicació, que permet guardar dades en format clau-valor. Per a gestionar fàcilment aquesta base de dades es va programar la classe estàtica *SharedPreferencesManager,* que defineix els mètodes de lectura, escriptura i esborrat de les dades.

Un cop es tenen carregades les dades bàsiques de l'usuari, s'inicia la vista de recomanacions d'usuaris, que és la primera finestra que es veu un cop dins l'app. Es demanen a l'API les seves recomanacions de perfils i a partir d'aquí se li dona llibertat perquè navegui pel sistema.

Per a navegar dins l'app s'han implementat dos menús ja mostrats en l'apartat de disseny. El primer, el lateral, és el menú estàndard del paradigma de disseny *Material Design* i se l'anomena *Navigation drawer* [41]. Des d'aquí es permet accedir a totes les funcionalitats considerades de segon nivell de l'app, tals com actualitzar el perfil o canviar la configuració del compte. El segon menú, l'inferior, se l'anomena *Bottom Navigation* [42] i permet accedir a les 3 funcionalitats principals: el xat, el cercador d'esdeveniments i la funcionalitat de recomanació de perfils. Aquests dos menús s'han implementat a la vista de l'InsideActivity, juntament amb el contenidor de fragments. D'aquesta manera, cada fragment que es vagi carregant disposarà d'aquests dos menús sense haver d'anar replicant la seva lògica. La figura 21 mostra els dos menús i marcat en blau el contenidor de fragments.

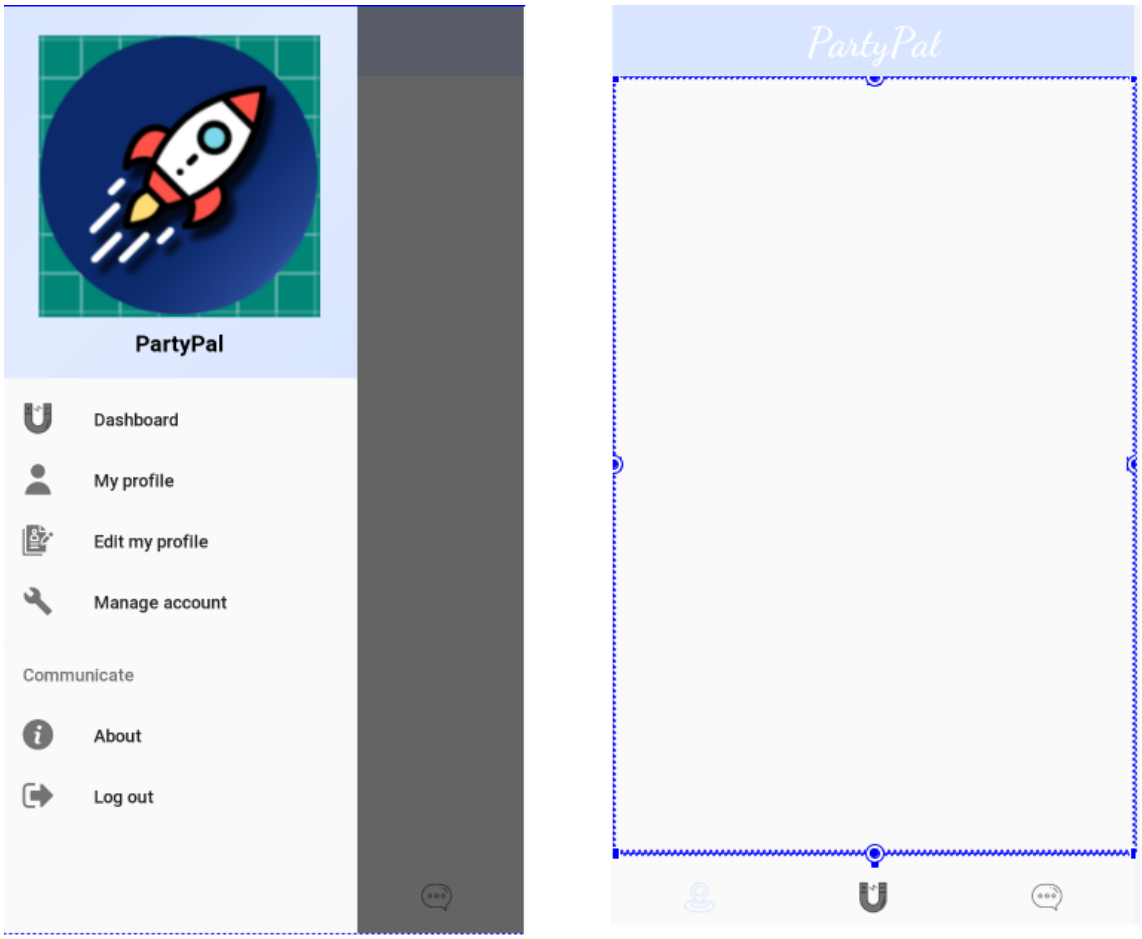


Figura 21: Menús disponibles dins l'app i contenidor de fragments

La gestió dels fragments actius i de la BackStack és responsabilitat de les dues activities. Com que aquestes no són les encarregades de mostrar les vistes, quan s'iniciïn el primer que faran serà determinar quin és el primer fragment que s'haurà de carregar i l'inflaran. Dins d'aquest fragment o quan s'interactuï amb algun dels menús i es demani navegar a un altre fragment, es realitzarà una crida al mètode *replaceFragment(fragment)* de l'activity pare passant una instància del fragment al què es vol saltar.

```
public void replaceAndDontDelete(Fragment fragment){
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.setCustomAnimations(R.anim.enter_from_right, R.anim.exit_to_left, R.anim.enter_from_left, R.anim.exit_to_right);
        fragmentTransaction.replace(R.id.fragment_container_inside, fragment, fragment.toString());
        fragmentTransaction.addToBackStack(fragment.toString());
        fragmentTransaction.commit();
    }
```

Figura 22: Codi de l'InsideActivity per a gestionar els fragments

La figura 22 mostra el codi. A part de la primera línia que crea l'objecte per a fer la transició d'un fragment a un altre, les tres línies importants són les següents:

- *fragmentTransaction.replace():* s'indica l'id del contenidor dels fragments en el que es vol realitzar la transició i el fragment nou que es vol posar.
- *fragmentTransaction.addToBackStack():* S'afegeix aquest fragment a la cima de la pila *BackStack*.
- *fragmentTransaction.commit():* S'executa la transició.

El fragment que es passa a aquesta funció no és directament la vista que veurà l'usuari; es passa la classe controlador que gestiona la seva lògica. Dins d'aquesta classe, la funció *onCreateView()* s'encarrega d'inflar la vista i tots els seus components a partir dels seus identificadors.

Per a transmetre informació entre fragments o entre activities s'han fet servir els objectes *Bundle*, que són petites instàncies a memòria local que permeten guardar dades en un format clau-valor. D'aquesta forma s'han pogut reutilitzar classes i fer-les genèriques, com és el cas del *ProfileFragment*, que s'encarrega de mostrar la informació relativa a un usuari. En un començament es van crear dues classes, una per al usuari que tenia la sessió iniciada, ja que es podrien agafar les dades de la *SharedPreferences*, i una altra per a qualsevol altre usuari, demanant les dades al servidor. Fent servir els *Bundles*, es va poder unificar tot en una mateixa classe, passant la informació necessària abans d'inflar la vista i deixant que fos la seva lògica la que decidís què fer. D'aquesta manera, s'ha seguit un plantejament similar al del patró d'estat [43], que permet canviar el comportament d'una classe en temps d'execució depenent del context extern.

8.1.3. Tecnologies usades i aplicades

Per al desenvolupament de l'app ha calgut treballat amb serveis tant propis com externs. A continuació es detalla el procés de comunicació i les eines que s'han fet servir.

Client d'inici de sessió de *Google*:

Tant al login com al registre es dona l'opció a l'usuari de connectar-se amb *Google* per sol·licitar la seva informació al servei i estalviar-se haver-la d'introduir manualment. En aquest cas *Google* ja ofereix un procediment estàndard per a realitzar la petició. Una adaptació del codi es pot veure a la figura 23.

```
private void initGoogleLogin() {
        GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestEmail()
                .build();
        mGoogleSignInClient = GoogleSignIn.getClient(getActivity().getApplicationContext(), gso);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == RC_SIGN_IN) {
            Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
            handleSignInResult(task);
        }
    }

    private void handleSignInResult(Task<GoogleSignInAccount> completedTask) {
        try {
            GoogleSignInAccount account = completedTask.getResult(ApiException.class);
            if (account.getEmail().length() != 0) user.setText(account.getEmail());

        } catch (ApiException e) {
            Log.w("Main Screen", "signInResult:failed code=" + e.getStatusCode());
        }
    }
```

Figura 23: Obtenció de les dades de l'usuari a partir de l'API de Google

La primera funció, *initGoogleLogin*(), infla la vista perquè l'usuari inicïi sessió o triï una de les que ja té obertes al dispositiu per agafar les dades. La segona, *onActivityResult*(), és un listener que s'activarà un cop es tanqui la vista iniciada per la funció anterior. Si el *requestCode* que retorna és el mateix que li havíem especificat nosaltres, voldrà dir que la crida ha estat un èxit i l'usuari ha proporcionat les dades requerides. La funció que realment fa el tractament de les dades és la tercera, *handleSignInResult*(), que transforma l'objecte retornat en un objecte operable i selecciona els camps que ens interessen, en aquest cas només el mail.

Firebase:

Firebase és una plataforma de serveis en el cloud gestionada per *Google* amb la finalitat de facilitar el procés de desenvolupament i la creació d'apps d'altes prestacions d'una forma ràpida. De totes les eines que proporciona, s'han fet servir dues: la base de dades en temps real i el sistema d'autenticació. La segona eina ha vingut dictada per requisits de la primera, atès que per fer servir gran part de la base de dades realtime, cal que hi hagi una instància activa d'un usuari, i aquest ha d'estar autenticat. Aquesta propietat d'en "temps real" de la base de dades ha estat crucial per a desenvolupar el sistema de xat en comparació amb una base de dades relacional. El fet que sigui realtime evita haver de realitzar un polling continu per a comprovar si hi ha nous missatges, aproximació inviable tan aviat com el nombre d'usuaris escalés mínimament.

```
private void readMessages(final String myid, final String userid, final String pictureBitmap){
    mChat = new ArrayList<>();
    reference = FirebaseDatabase.getInstance().getReference( path: "Chats");
    reference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            mChat.clear();
            for (DataSnapshot snapshot: dataSnapshot.getChildren()){
                Chat chat = snapshot.getValue(Chat.class);
                assert chat != null;
                if (chat.getReceiver().equals(myid) && chat.getSender().equals(userid) ||
                chat.getReceiver().equals(userid) && chat.getSender().equals(myid)){
                    mChat.add(chat);
                }
                messageAdapter = new MessageAdapter(getContext(),mChat, pictureBitmap);
                recyclerView.setAdapter(messageAdapter);
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });
}
```

Figura 24: Gestió de la lectura de missatges de Firebase

La figura 24 mostra el codi per a realitzar la lectura d'un xat de la base de dades de *Firebase*. Aquesta funciona aplicant una adaptació del patró observador [44]. Quan sol·licitem llegir una dada, aquesta no ens és retornada de forma immediata; és per això que programem un listener, que estarà escoltant fins que arribi. A diferència del patró observador, no ens notificarà amb un avís indicant que les dades han canviat, sinó que ens enviarà les dades en si. Cada cop que hi hagi un canvi en aquest valor o grup de la base de dades, aquesta enviarà les dades a tots els dispositius que n'estiguin subscrits, és a dir, que tinguin un listener apuntant a aquesta referència.

Aquest comportament descrit es pot observar a la segona línia de codi, on es demana la referència de la taula de "Chats" per a la línia següent programar el listener.

APICommunicator:

Encara que des de l'app s'ha accedit a serveis externs com *Firebase*, la principal comunicació que s'ha fet amb l'exterior ha estat amb el nostre servidor. Per a estandarditzar aquesta comunicació, es va programar la classe *APICommunicator*, que ofereix els mètodes *getRequest(), postRequest(), putRequest() i deleteRequest()*, coincidint amb els permesos a l'API. Des de qualsevol part de l'app que requerís dades o processament d'aquestes per part del servidor, es realitzava una crida similar a la mostrada a la figura 25.

```
private void cridaAPI(T paramsNecessaris) {
    APICommunicator apiCommunicator = new APICommunicator();
    Response.Listener responseListener = new Response.Listener<CustomRequest.CustomResponse>() {
        @Override
        public void onResponse(CustomRequest.CustomResponse response) {
            //codi si la crida retorna correcte
        }
    };
    Response.ErrorListener errorListener = new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            //codi si la crida retorna error
            errorTreatment(error.networkResponse.statusCode);
        }
    };
    Map<String, Object> params = new HashMap<>();
    params.put(paramsNecessaris);
    apiCommunicator.postRequest(getActivity().getApplicationContext(), Url, responseListener, errorListener, params);
}
```

Figura 25: Invocació de la crida a l'API i gestió del resultat

La figura 24 mostra com abans de realitzar la invocació a la crida (última línia), cal preparar els listeners per gestionar el resultat que retorni. Cal programar-ne dos: un pel cas de crida correcta i l'altre pel cas d'error. També cal empaquetar les dades que s'enviaran al servidor; es fa servir l'objecte *params*. Com en aquest cas la crida és un post, dins de la classe *APICommunicator* es cridarà la funció de la figura 26.

```
void postRequest(Context context, String url, Response.Listener responseListener, Response.ErrorListener errorListener,
final Map<String, Object> params) {
    doRequest(context, Request.Method.POST, url, responseListener, errorListener, params);
}
```

Figura 26: Crida dins la classe APICommunicator per a gestionar les peticions post.

Per últim, la figura 27 mostra la funció que realment fa la crida. Es fa servir la llibreria *Volley* per a gestionar la cua de peticions i la comunicació HTTP.

```
private void doRequest(final Context context, final int method, final String url,
final Response.Listener responseListener, final Response.ErrorListener errorListener, final Map<String, Object> params) {
    CustomRequest request = new CustomRequest(method, API_URL + url, responseListener, errorListener) {
        @Override
        public byte[] getBody() {
            try {
                return new JSONObject(params).toString().getBytes(CHARSET);
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
                return null;
            }
        }

        @Override
        public String getBodyContentType() {
            return CONTENT_TYPE;
        }
    };
    request.setRetryPolicy(new DefaultRetryPolicy(
            1000,
            4,
            DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
    request.setShouldCache(false);
    Volley.newRequestQueue(context).add(request);
}
```

Figura 27: Crida a l'API fent servir la llibreria Volley.

La part més important del codi de la figura 27 és *request.setRetryPolicy(),* on s'especifica el temps en milisegons per a donar per perduda una crida (en aquest exemple, 1 segon), el nombre màxim de reintents en cas de fallada (en aquest exemple 4) i el multiplicador de retard, que serveix per augmentar el temps d'espera entre crida i crida de forma exponencial (en aquest exemple s'ha fet servir DEFAULT_BACKOFF_MULT, que és 1, és a dir, sense augment).

Per tal d'agilitzar la comunicació i a la vegada gastar menys dades de la tarifa del client, quan ha calgut moure imatges entre el servidor i l'app i a l'inrevés, aquestes han estat comprimides

prèviament. Per fer això es va programar una classe anomenada *ImageCompressor*, que passa les imatges del format Bitmap en el que treballa Android a una String, fent la compressió en base 64. D'aquesta manera, les imatges poden ser carregades i descarregades com si es tractés d'un text, facilitant molt la gestió de recursos.

A la vegada, com totes les dades que retornen del servidor venen en format JSON, un cop arriben a l'app cal convertir-les al format necessari perquè puguin ser d'utilitat. És per això que ha calgut programar adaptadors. Un adaptador agafa un conjunt de dades i les mostra d'una manera determinada depenent de com estigui programat. En el cas de l'adaptador del sistema de xat, per exemple, per a cada missatge nou que arribava, determinava si l'emissor era el mateix usuari o la persona amb qui estava parlant, i depenent d'això pintava d'un color o d'un altre la bombolla on s'inseria el text. Han calgut adaptadors per al llistat d'esdeveniments, el llistat de xats que té un usuari, la mateixa funcionalitat del xat (intercanvi de missatges) i pel visualitzador del perfil de l'usuari.

SharedPreferencesManager:

Encara que no és un servei extern com a tal, per a gestionar la base de dades local, la *SharedPreferences*, s'ha creat la classe *SharedPreferencesManager,* que funciona com API. Aquesta classe segueix el patró de disseny repositori, que tal com descriu la guia de desenvolupadors de Microsoft [45], "Els repositoris són classes o components que encapsulen la lògica necessària per tenir accés a fonts de dades. Centralitzen la funcionalitat d'accés a dades comunes, el que proporciona un millor manteniment i el desacoblament de la infraestructura o tecnologia que s'usa per accedir a bases de dades des del nivell de model de domini." La classe implementa tres funcionalitats: escriptura, lectura i esborrat a la *SharedPreferences*.

```
fun remove(context: Context, key: String) {
    val sharedPref = context.getSharedPreferences(PREFERENCES_FILE, Context.MODE_PRIVATE)
            ?: return
    with(sharedPref.edit()) { this: SharedPreferences.Editor!
        remove(key)
        apply()
    }
}
```

Figura 28: Operació d'esborrat de la SharedPreferences dins la classe SharedPreferencesManager

La figura 28 mostra l'operació d'esborrat de la *SharedPreferencesManager* escrita en Kotlin. El primer que es fa és localitzar la *SharedPreferences* a partir de la ruta definida a la variable *PREFERENCES_FILE* i obrir-la en mode privat, que vol dir que qualsevol dada que s'escrigui en aquest fitxer només podrà ser accedida per l'aplicació que l'ha creat i no per qualsevol altra instal·lada al dispositiu. El següent que es fa és cridar a la funció *remove()* de la interfície pare de la *SharedPreferences* i seguidament aplica el canvi cridant a *apply().*

Ja que la SharedPreferences és una memòria interna, no té una interfície gràfica per a gestionar les dades que emmagatzema. És per aquest motiu que s'ha fet servir la llibreria Stetho[46] desenvolupada per *Facebook*. Aquesta no permet modificar, però si visualitzar les dades guardades.

Llibreries

A part de Stetho, s'han fet servir altres llibreries per ajudar-nos a l'hora de programar l'aplicació:

-Google design: incorpora tots els complements i widgets oficials amb estil Material Design.

-Volley: permet realitzar crides a serveis externs i s'encarrega de gestionar el transport en HTTP.

-Carouselview: proporciona la vista de fotografies en format control lliscant usat al visualitzador del perfil de l'usuari.

-Gson: permet gestionar les dades en format JSON i transformar-les en objectes del sistema fàcilment.

-Circleimageview: permet transformar una imatge qualsevol en a una rodona.

-Swipecards: proporciona l'efecte de moure les targetes a dreta o esquerra de la funcionalitat de recomanació de perfils.

8.2. Implementació servidor (back end)

8.2.1. Spring Boot

El codi del servidor ha estat desenvolupat fent servir el framework *d'Spring Boot*. *Spring Boot* és una infraestructura lleugera que elimina la major part de la feina de configurar els projectes basats en *Spring*.

Spring

Spring és un framework de codi obert per a la plataforma Java basat en el patró de disseny de la injecció de dependències. Aquest "és un patró de disseny orientat a objectes, en què se subministren objectes a una classe en lloc de ser ella mateixa qui els creï. Aquests objectes compleixen contractes que necessiten les nostres classes per poder funcionar (d'aquí el concepte de dependència). Les nostres classes no creen els objectes que necessiten, sinó que se'ls subministra una altra classe 'contenidora' que injectarà la implementació desitjada al nostre contracte" [47]. L'objectiu d'aquest patró és crear components altament reutilitzables i modulars, a la vegada que s'intenta reduir al màxim la dependència (o acoblament) entre ells.

Per a entendre aquest patró es representarà gràficament amb les figures 29 i 30.

```
public class A {
    private B dependency;
    public A(){
        dependency = new B();

    }

}
```

```
class Alumne{
    private Calificacions calificacions;
    private Aula aula;
    private Info info;
    constructor(){
        this.calificacions = Calificacions.all();
        this.aula = Aula.find(this.id);
        this.info = new StudentInfo();
    }
}
```

Figura 29: Exemples de codi sense aplicar injecció de dependències

La figura 29 mostra dos exemples de codi on no s'aplica el patró d'injecció de dependències. En el primer exemple, la creadora de la classe A crea una instància de la classe B per associar-la a la variable interna de la mateixa classe; en el segon exemple, la constructora coneix com accedir a la informació que necessita de les altres classes. En qualsevol dels dos mètodes, es crea un acoblament alt, fet que dificulta un possible canvi en la implementació interna de les classes.

La solució a aquest problema es pot veure a la figura 30, on aplicant el patró de la injecció de dependències, es delega la creació de l'objecte requerit dins la crida a l'objecte que l'ha invocat.

```
public class A {
    private B dependency;
    public A(B instancedepency){
        this.dependency=instancedepency;
    }
}
```

```
class Alumne{
    private Calificacions calificacions;
    private Aula aula;
    private Info info;
    constructor(Calificacions calificacio, Aula aula, Info info){
        this.calificacions = calificacio;
        this.aula = aula;
        this.info = info;
    }
}
```

Figura 30: Exemples de codi aplicant injecció de dependències

La popularitat d'*Spring* sorgeix, entre altres motius, per la possibilitat d'automatitzar aquest procés a partir dels *beans*. Els *beans*, activats a partir del tag *@Autowired*, funcionen com a identificadors o etiquetes. En temps de compilació, el compilador busca les etiquetes i les connecta amb la seva dependència de forma automàtica. D'aquesta manera, cadascun dels camps de la figura 31 ha estat inicialitzat sense necessitat de referenciar a l'objecte original, d'això se n'ha encarregat el framework.

```
@Autowired
private EventRepository eventRepository;

@Autowired
UserRepository userRepository;

private void saveEvents(){
    eventRepository.saveEvent(EventTestUtils.createEvent("G5vYZ4JVkma8o"));
    eventRepository.saveEvent(EventTestUtils.createEvent("k7vGF4NUlsxSd"));
    eventRepository.saveEvent(EventTestUtils.createEvent("Z698xZ2qZadsD"));
}
```

Figura 31: Injecció de dependències a partir de la propietat @Autowired

Altres característiques interessants que ofereix *Spring* són la programació orientada a aspectes, el suport a l'accés a dades usant Java Database Connectivity (JDBC), mòduls per a l'autenticació d'usuaris, possibilitat d'implementació el Model-Vista-Controlador...

Spring Boot

Spring Boot és un framework de codi obert basat en Java usat per a desenvolupar micro serveis i construir projectes *Spring* independents.

Els micro serveis són un tipus d'arquitectura de software que permet als programadors desenvolupar i desplegar serveis de manera independent. Cadascun d'aquests serveis té el seu propi procés i això aconsegueix crear solucions lleugeres i fàcilment escalables per donar suport a les aplicacions empresarials. La idea es basa en separar la funcionalitat en petites aplicacions independents que puguin operar amb completa autonomia, les quals exposin la seva

funcionalitat com a serveis, amb la idea que siguin reutilitzables. La figura 32 mostra aquest comportament.

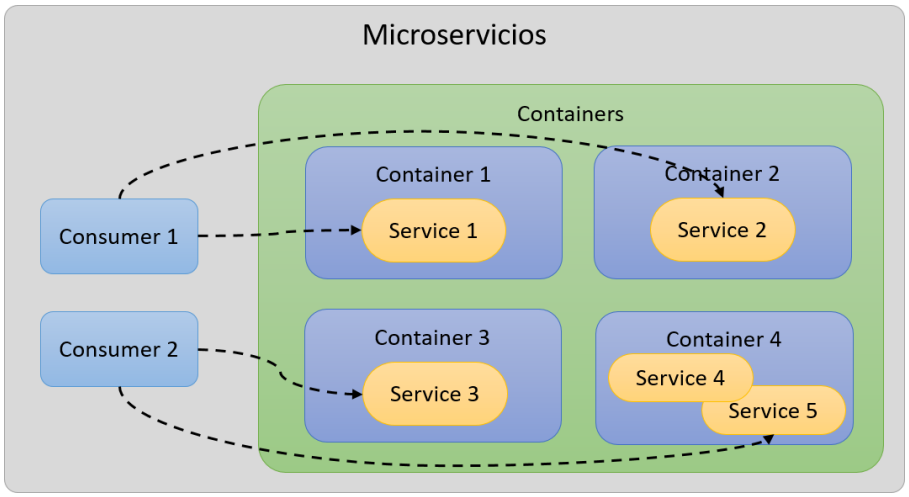


Figura 32: Esquema conceptual del funcionament dels micro serveis. Font: oscarblancarteblog.com

Els micro serveis ofereixen els següents avantatges als desenvolupadors: fàcil implementació, escalabilitat senzilla, compatibilitat amb contenidors, mínima configuració requerida i necessitat d'un temps de producció menor.

Spring Boot proporciona una plataforma per a crear projectes en *Spring* estalviant tot el procés de configuració previ (configuració inicial del projecte i especificació manual de dependències) i de desplegament en servidor, permetent que el programador es pugui centrar en el procés important, que és desenvolupar el servei. Per això es diu que les aplicacions en Spring Boot són "just run", ja que el temps requerit d'ençà que es comença el projecte fins que es pot fer el primer "run" és molt menor que fent servir altres frameworks.

Altres característiques importants que ofereix *Spring Boot* són les següents:

- Proporciona una manera flexible de configurar Java Beans, configuracions XML i transaccions de base de dades.
- Proporciona un potent processament per lots i gestiona els endpoints de REST.
- Alta configuració automàtica; no es necessiten configuracions manuals.
- Ofereix les anotacions o etiquetes ofertes per *Spring*.
- Afavoreix la gestió de les dependències.
- Inclou contenidor de servlet incorporat.

8.2.2. Configuració del servidor

Per a allotjar el servei s'ha llogat un servidor a *DigitalOcean*. La màquina disposa d'1 GB de ram, 25 GB de disc i corre sobre un sistema operatiu Ubuntu 18.04.2 versió 64 bits. Per reduir la latència entre clients i servidor, aquest està ubicat a Frankfurt. En cas que calgui paral·lelitzar l'aplicació per escalar-la, s'estudiarà a quina part del món situar el nou servidor. S'ha instal·lat la versió de Java 8.

Dins del servidor, el sistema d'integració contínua (explicat més endavant) s'encarrega de desplegar el programa en la seva última versió cada cop que aquesta canvia. Per a la base de dades, s'ha creat un contenidor *Docker* amb una imatge *Postgre* versió 11.

El programa es va pensar per a funcionar com a API i més específicament com a servei REST (Representational State Transfer). De forma resumida, "REST és qualsevol interfície entre sistemes que utilitzi HTTP per obtenir dades o generar operacions sobre aquestes en tots els formats possibles, però XML i JSON són els més estàndards" [48]. En el nostre cas, tot retorn de dades és en format JSON.

Algunes de les característiques de REST són les següents:

- Client-Servidor: aquesta restricció manté al client i al servidor dèbilment acoblats. Això vol dir que el client no necessita conèixer els detalls d'implementació del servidor i el servidor es "despreocupa" de com són usades les dades que envia al client.
- Sense estat: cada petició HTTP conté tota la informació necessària per executar-la, el que permet que ni client ni servidor necessiten recordar cap estat previ per satisfer-la.
- Les operacions més importants són el POST, el GET, el PUT i el DELETE.
- Els objectes en REST sempre es manipulen a partir de la URI: la URI és l'element identificador únic per accedir als recursos del sistema.

Spring Boot ofereix etiquetes per a facilitar el desenvolupament en REST. La figura 33 mostra una de les funcions dins del controlador d'usuaris. La primera línia determina que l'operació espera una crida de tipus POST sota la URI /users. La segona determina la resposta en cas que la funció *createUser()* no retorni cap excepció. El propi framework pot convertir el cos de la crida en un objecte del sistema si es defineix una classe amb atributs interns amb els mateixos noms als dels atributs de la crida. En aquest cas, s'ha convertit automàticament el cos de la crida a una instància de l'objecte User.

```
@PostMapping("/users")
@ResponseStatus(value = HttpStatus.CREATED, reason = "User succesfully created")
public void createUser(@RequestBody User user) {
    userRepository.saveUser(user);
}
```

Figura 33: Exemple d'ús de les etiquetes d'Spring Boot per a crear un servei REST

8.2.3. Implementació de les funcionalitats principals

Gran part de la gestió del sistema de xat es va dur a terme a l'app, per tant es procedeix a detallar com han estat implementades les altres funcionalitats principals.

Login i registre:

La funcionalitat de registre d'usuaris es va implementar com qualsevol de les altres funcionalitats del sistema back end: es va crear un endpoint amb l'etiqueta corresponent i es va lligar a una operació que s'encarregava d'introduir les dades a la base de dades. Per a la funcionalitat del login, però, es va fer servir un dels mòduls que ofereix el propi framework.

Per fer servir aquest mòdul cal definir una classe que hereti de *WebSecurityConfigureAdapter*. Dins d'aquesta, cal redefinir l'operació *configure*(), que indicarà quin tractament s'ha de realitzar a cadascuna de les peticions http que arribin. Aquest tractament consisteix en filtres pels quals ha de passar la petició per a determinar si és vàlida o no. S'han definit dos filtres: *JWTAuthenticationFilter,* que hereta de *UsernamePasswordAuthenticationFilter* i *JWTAuthorizationFilter,* que ho fa de *BasicAuthenticationFilter*. El primer s'encarrega de gestionar el login dels usuaris i el segon de comprovar que en cada crida que es realitza a alguna

URI d'un objecte que requereixi que l'usuari tingui una sessió activa (és a dir, totes les funcionalitats de l'interior de l'app), s'adjunti un token vàlid. El token es retorna quan el *JWTAuthenticationFilter* determina que les credencials són vàlides i és emmagatzemat en memòria local del dispositiu Android fins que es tanqui la sessió o caduqui el token.

Cercador d'esdeveniments:

Per implementar la funcionalitat del cercador, es va fer servir l'API de *Ticketmaster*. Per a poder usar-la calia crear un compte d'usuari i sol·licitar una APIKey. Aquesta clau, igual que el token proporcionat a un usuari quan inicia sessió en el nostre sistema, cal guardar-la i enviar-la en cada petició. En cas que no sigui correcta, el sistema de *Ticketmaster* retorna error.

L'API del servei és molt extensa i permet una àmplia configuració pel que fa a cercar a partir d'etiquetes que s'afegeixen al path de la crida. En el nostre cas, només s'ha fet servir la cerca per paraula clau i per localització de l'esdeveniment.

El que no és tan personalitzable és el format de retorn: el servei retorna tota la informació de la qual disposa independentment que només necessitem un parell de camps, fet que obliga a fer el tractament de les dades de forma manual. Per a fer-ho, va caler crear una classe que contingués tots els atributs que es tornaven a la crida. Com la crida feia servir un format JSON i aquest permet anidar atributs dins d'atributs, va caler crear tot un entramat de classes que es referenciaven les unes a les altres per aconseguir fer la conversió completa. Un cop les dades es tenien al servidor, se seleccionava la informació necessària i es creava un paquet que es retornava a l'app.

Algoritme de recomanacions:

L'algoritme de recomanacions s'activa tan bon punt l'usuari entra a l'app. El procediment que segueix és el següent:

El primer pas és crear una estructura que ens permeti afegir entrades dinàmicament o modificar-les. Per a representar aquesta estructura s'ha fet servir un diccionari. Cada entrada tindrà com a clau un nom d'usuari i com a valor la seva puntuació. Seguidament, per cada esdeveniment que segueix l'usuari que ha activat l'algoritme s'agafen tots els usuaris que estan subscrits i es mira que no siguin ja match amb ell o ella, que no els hagi guardat o que no els hagi reportat. Si no compleixen cap d'aquests punts, s'afegeixen al diccionari amb una puntuació d'un o si ja són presents, s'incrementa en un el valor de la seva puntuació.

En acabar d'iterar aquesta bossa d'esdeveniments, el resultat és el diccionari omplert amb la quantitat d'esdeveniments compartits entre un usuari qualsevol i el nostre usuari. D'aquesta forma i aprofitant l'ordenació per valor, l'entrada que tingui una puntuació més alta serà la que tindrà uns gustos més similars i, per tant, el primer perfil que se li recomanarà. Abans de retornar les dades, es filtra pels interessos definits per l'usuari que ha activat l'algoritme, és a dir, si ha indicat que només li interessen homes, per exemple, es descartaran tots els altres perfils. Un cop fet això, es crea una estructura on es van introduint les dades que es volen retornar dels usuaris que han passat el filtre. Aquesta estructura és la que se li retorna al front end perquè infli les vistes.

8.2.4. Base de dades

La base de dades del sistema està allotjada en un contenidor *Docker* construït a partir d'una imatge *PostgreSQL* versió 11, per tant es farà servir un model relacional. Per accedir a la base de dades es determinen un seguit de ports on el contenidor estarà escoltant peticions. Aquests estan oberts només per accés local i no accepten peticions http, per tant l'única porta d'entrada a la base de dades és a partir del programa back end allotjat al servidor.

Per automatitzar la construcció del contenidor s'ha creat un arxiu *docker-compose.yml*, que permet definir la imatge, versió, ports d'entrada i sortida del contenidor per tal de centralitzar la seva configuració i permetre la seva construcció a partir d'una comanda d'una línia, que s'executa cada cop que el sistema d'integració contínua actualitza la versió del sistema que està funcionant.

Per a construir la base de dades es va fer servir el sistema de migracions, que permet definir-la de forma incremental a partir d'arxius versionats. D'aquesta manera, si en un moment donat es vol crear una classe Usuaris, es crearà un arxiu V1__crear_taula_usuaris.sql amb el codi sql dins. Si en algun altre moment futur es volgués modificar aquesta taula, no caldria eliminar-la del sistema, només s'hauria de crear l'arxiu V2__modificar_taula_usuaris.sql amb els canvis desitjats.

L'estat final de la base de dades és el següent:

```
CREATE TABLE "user"
(
  ID VARCHAR(255) NOT NULL;
  email VARCHAR(255) PRIMARY KEY,
  name  VARCHAR(255) NOT NULL,
  lastName VARCHAR(255) NOT NULL,
  birthdate DATE NOT NULL,
  gender VARCHAR(255) NOT NULL,
  password VARCHAR(255) NOT NULL,
  interests VARCHAR(255) NOT NULL,
  picture TEXT,
  recovery VARCHAR(255) DEFAULT NULL,
picture2 TEXT DEFAULT NULL,
picture3 TEXT DEFAULT NULL,
bio TEXT DEFAULT NULL
);
```

```
CREATE TABLE "event"
(
   id VARCHAR(255) PRIMARY KEY,
   name  VARCHAR(255) NOT NULL,
   url VARCHAR(255) NOT NULL,
   eventVenue VARCHAR(255),
   eventCity VARCHAR(255) NOT NULL,
   eventLocalDate VARCHAR(255) NOT NULL,
   eventLocalTime VARCHAR(255),
   picture1 TEXT,
   picture2 TEXT,
   picture3 TEXT
);
```

```
CREATE TABLE "Subscriptions"
(
   userID  VARCHAR(255),
   eventID VARCHAR(255),
   PRIMARY KEY (userID, eventID),
   FOREIGN KEY (userID) REFERENCES "user"
(email),
   FOREIGN KEY (eventID) REFERENCES
"event" (id)
);
```

```
CREATE TABLE "match" (
        user1 VARCHAR(255),
        user2 VARCHAR(255),
        primary key (user1, user2)
)
```

```
CREATE TABLE "pending" (
        user1 VARCHAR(255),
        user2 VARCHAR(255),
        primary key (user1, user2)
)
```

8.3. Implementació integració continua

"La integració contínua és una pràctica de desenvolupament de programari mitjançant la qual els desenvolupadors combinen els canvis en el codi en un repositori central de forma periòdica, permetent l'execució de versions i proves automàtiques. La integració contínua es refereix majoritàriament a la fase de creació o integració del procés de publicació de software"[49].

Per a la nostra aplicació, el sistema d'integració continua *CirlceCI* s'activa tan bon punt actualitzem el codi del repositori del servidor a *GitHub*. El primer pas que realitza és una còpia d'aquest codi i el porta als seus servidors, on construeix el projecte i executa tots els tests que aquest conté. Si cap falla, agafa el paquet .jar resultant que conté l'aplicació i l'envia via SCP al nostre servidor. Un cop el paquet ha arribat de forma completa, es reinicia el contenidor *Docker* de la base de dades per evitar que es quedi cap crida en un estat inconsistent en el transcurs del procés d'actualització del servidor. Per últim, s'executa un servei bash que s'ha definit per a aturar la versió anterior del programa, eliminar-la i iniciar-ne la nova.

Tot aquest procés cal definir-lo en el fitxer config.yml en una carpeta dins del nostre repositori de codi. En aquest arxiu es defineixen els passos que volem que executi el sistema d'integració contínua.

Si quan el sistema d'integració està construint el codi o executant els tests algun procés falla, atura la construcció i no la desplega en el servidor de producció, deixant la versió actual. També ens avisa pel mitjà que especifiquem (mail en el nostre cas) que hi ha hagut un error i informació sobre aquest. D'aquesta manera el temps entre que es produeix un i el que triga a assabentar-se el programador o gestor del projecte és mínim.

9. Validació i proves

Un cop implementat el sistema, queda comprovar que s'han satisfet tots els requeriments definits a l'apartat 5. Anàlisi de requisits: tant funcionals com no funcionals. Per donar com a satisfets els requisits funcionals s'han fet servir dues metodologies: tests automatitzats i tests o validació manual. Els tests automatitzats s'aplicaran al component del servidor i els manuals al component de l'app mòbil, on automatitzar tests és molt més complicat i a més tindrien un cicle de vida molt curt perquè tan aviat s'actualitzés el sistema operatiu, s'haurien de canviar.

Dins dels tests automatitzats s'ha treballat amb tests unitaris i tests d'integració [50]. Els primers són proves de molt baix nivell, properes al nucli de l'aplicació. Consisteixen en provar mètodes i funcions individuals de les classes, components o mòduls utilitzats pel codi del programa. Els segons comproven que els diferents mòduls o serveis utilitzats per l'aplicació funcionen bé junts.

Pel que fa a als tests manuals, per cada cas d'ús es provaran diferents escenaris per determinar que realment funciona correctament i no només en el cas idíl·lic. Els tests manuals són de gran utilitat per descobrir nous errors, perquè permeten simular amb molta precisió un escenari real, on els usuaris salten de vista en vista i proven les funcionalitats de l'aplicació, oblidant-se de la rigorositat que pot tenir un test formal.

Per l'altra banda, per donar com a satisfets els requisits no funcionals, es determinarà a partir del compliment dels criteris de satisfacció de cada requisit.

9.1. Tests automatitzats dels requisits funcionals

Per aquest projecte s'ha intentat cobrir la major part del codi del servidor a partir de diferents tests. S'ha fet servir la llibreria JUnit4 i juntament amb algunes funcionalitats del propi framework *s'Spring Boot*, s'han programat un total de 69 tests. Aquests s'han organitzat per blocs depenent del model o la funcionalitat clau que intentaven cobrir.

Per a intentar minimitzar el marge d'error, s'ha intentat triangularitzar les funcionalitats principals, és a dir, s'han provat diferents escenaris per assegurar que no només funciona en el cas idíl·lic.

La taula 8 resumeix com s'han repartit aquests 69 tests. És convenient recordar que perquè una versió sigui desplegada en producció, cal que es passin tots ells.

Tipus de test	Nombre de tests
Tests unitaris als filtres del sistema de login	2
Tests d'integració al sistema de login	3
Tests unitaris a les funcionalitats internes del sistema de registre	2
Tests d'integració al sistema de registre	4
Tests d'integració per a l'obtenció d'usuaris	3
Tests d'integració al sistema de recuperació de contrasenya	1
Tests d'integració per a la gestió dels perfils d'usuaris	3
Tests d'integració per a la configuració del compte	6

Tests unitaris per a la gestió de l'algoritme de recomanació	12
Tests d'integració per a l'algoritme de recomanació	5
Tests unitaris per a la gestió de la cerca i subscripció a esdeveniments	18
Tests d'integració per a la cerca i subscripció d'esdeveniments	10

Taula 8: Resum de la composició dels tests en el projecte

Per al component front end no s'han programat tests, ja que s'ha seguit una metodologia prova-error. Atès que la lògica que pot existir en aquest component s'encarrega només de gestionar com es veu la vista, trobar que alguna part no funciona és relativament fàcil. Per a comprovar que el comportament de l'app és realment el desitjat, s'ha fet servir l'emulador del propi entorn de treball, que permet virtualitzar dispositius de les característiques desitjades. En el nostre cas s'han provat diferents mides i resolucions de pantalla.

9.2. Validació manual dels requisits funcionals

Per a estandarditzar la prova de cada cas d'ús es farà servir la taula amb els camps que es defineixen a continuació:

- **Escenari**: defineix la prova i les circumstàncies en què es realitza.
- **Resultat**: resultat final de la prova. Hi haurà dues opcions: èxit, fallada.
- **Observacions del resultat**: comentaris o observacions que hagin sorgit de la realització de la prova.

#1 Log in		
Escenari	**Resultat**	**Observacions**
S'ha provat accedir al sistema amb unes credencials vàlides	Èxit	L'usuari s'ha autenticat i se l'ha redirigit a l'interior de l'aplicació
S'ha provat accedir al sistema amb unes credencials invàlides	Èxit	No es permet l'accés de l'usuari a l'interior de l'aplicació i se l'avisa que les credencials no són vàlides

#2 Log in amb *Google*		
Escenari	**Resultat**	**Observacions**
S'ha sol·licitat accedir al sistema amb les credencials de *Google*	Èxit	El camp del correu s'omple automàticament i només cal que l'usuari introdueixi la contrasenya
S'ha sol·licitat accedir al sistema amb unes credencials de *Google* invàlides	Èxit	El camp del correu no s'omple i se li indica a l'usuari que les dades introduïdes no existeixen

#3 Log out		
Escenari	**Resultat**	**Observacions**
L'usuari es troba dins l'aplicació i sol·licita tancar la seva sessió	Èxit	La sessió de l'usuari es tanca i se'l redirigeix a la vista de login

#4 Registre		
Escenari	**Resultat**	**Observacions**
L'usuari omple les dades sol·licitades al formulari de registre i demana crear el compte	Èxit	Es crea el compte d'usuari i es redirigeix aquest a la vista de login
L'usuari indica com a clau primària de compte (mail) un correu ja en ús	Èxit	No es crea el compte i s'indica a l'usuari que ja existeix un compte vinculat a aquest correu

#5 Registre amb *Google*		
Escenari	**Resultat**	**Observacions**
L'usuari sol·licita registrar-se amb *Google* amb un compte vàlid	Èxit	Determinats camps del formulari s'omplen de forma automàtica
L'usuari sol·licita registrar-se amb *Google* amb un compte invàlid	Èxit	No s'omple cap camp del formulari i se li indica a l'usuari l'error

#6 Recuperació de contrasenya		
Escenari	**Resultat**	**Observacions**
L'usuari vol recuperar la contrasenya del seu compte i introdueix una clau primària vàlida i després enganxa a l'app un codi de recuperació vàlid	Èxit	S'envia al mail indicat per l'usuari un codi de recuperació i quan l'introdueix a l'app juntament amb la nova contrasenya, aquesta s'actualitza
L'usuari vol recuperar la contrasenya del seu compte però introdueix una clau primària invàlida o no registrada al sistema	Èxit	S'indica a l'usuari que la clau primària no existeix o no és vàlida i no es prossegueix amb el procés de recuperació
L'usuari vol recuperar la contrasenya del seu compte i introdueix una clau primària vàlida i després enganxa a l'app un codi de recuperació invàlid	Èxit	S'indica a l'usuari que el codi de recuperació no existeix i no es prossegueix amb el procés de recuperació

#7 Buscar esdeveniment per nom al buscador		
Escenari	**Resultat**	**Observacions**
L'usuari introdueix el nom (i la ciutat opcionalment) de l'esdeveniment que desitja buscar i que existeix al sistema	Èxit	Es retorna a l'usuari un o més esdeveniments que coincideixen amb la cerca realitzada
L'usuari introdueix el nom (i la ciutat opcionalment) de l'esdeveniment que desitja buscar que no existeix al sistema	Èxit	S'indica a l'usuari que la seva cerca no coincideix amb cap entrada del sistema i no es retorna cap esdeveniment

#8 Seguir un esdeveniment		
Escenari	**Resultat**	**Observacions**
L'usuari troba un esdeveniment que li interessa i sol·licita seguir-lo	Èxit	Es registra que l'usuari segueix aquest esdeveniment al sistema i s'actualitza la vista d'aquest

#9 Deixar de seguir un esdeveniment		
Escenari	**Resultat**	**Observacions**
L'usuari vol deixar de seguir un esdeveniment i ho indica al sistema	Èxit	Es registra que l'usuari deixa de seguir aquest esdeveniment al sistema i s'actualitza la vista d'aquest

#10 Llistar els esdeveniments que segueixes		
Escenari	**Resultat**	**Observacions**
L'usuari accedeix a la finestra d'esdeveniments	Èxit	El sistema llista els esdeveniments que l'usuari segueix, si en té cap

#11 Visualitzar un esdeveniment		
Escenari	**Resultat**	**Observacions**
L'usuari vol veure la informació relativa a un esdeveniment i prem sobre ell	Èxit	El sistema mostra tota la informació que té respecte a l'esdeveniment sol·licitat

#12 Mostrar usuaris recomanats		
Escenari	**Resultat**	**Observacions**
L'usuari vol veure les recomanacions d'altres perfils que li fa el sistema i accedeix a aquesta funcionalitat	Èxit	El sistema li va recomanant perfils basant-se en els esdeveniments que segueix

#13 Enviar un missatge a un usuari que ens interessa		
Escenari	**Resultat**	**Observacions**
Fent servir la funcionalitat de les recomanacions, l'usuari s'interessa per un perfil i li envia un missatge	Èxit	L'usuari receptor u2 rep el missatge i l'usuari emissor u1 ja no pot interactuar amb u2 fins que u2 accepti la sol·licitud de match, si ho fa

#14 Rebutjar un usuari que no ens interessa		
Escenari	**Resultat**	**Observacions**
Fent servir la funcionalitat de les recomanacions, el sistema mostra un perfil a l'usuari que no li interessa i el rebutja	Èxit	El sistema recomana el següent usuari a la llista de recomanacions, si encara en queda algun

#15 Guardar un usuari que ens pot interessar		
Escenari	**Resultat**	**Observacions**
Fent servir la funcionalitat de les recomanacions, el sistema mostra un perfil a l'usuari que li interessa i el guarda	Èxit	El sistema registra que a l'usuari u1 ha guardat a l'usuari u2 i resta a l'espera de la interacció de l'usuari u2

#16 Navegar pel perfil d'un usuari		
Escenari	**Resultat**	**Observacions**
L'usuari està fent servir l'algoritme de recomanació i vol navegar pel perfil de la recomanació actual	Èxit	El sistema li mostra nom, cognom, bio i fotos relatives a l'usuari que li ha recomanat

#17 Respondre a un missatge de match		
Escenari	**Resultat**	**Observacions**
L'usuari rep un missatge de match i el respon	Èxit	S'envia el nou missatge i es registra el match al sistema

#18 Rebutjar un missatge de match		
Escenari	**Resultat**	**Observacions**
L'usuari rep un missatge de match i el rebutja	Èxit	S'elimina el xat de la llista de xats de l'usuari receptor i no es registra el match

#19 Llistar xats		
Escenari	**Resultat**	**Observacions**
L'usuari accedeix a la funcionalitat per a llistar els xats	Èxit	El sistema llista els xats que té amb altres usuaris

#20 Xatejar amb un usuari que ja és match		
Escenari	**Resultat**	**Observacions**
L'usuari u1 intercanvia missatges amb l'usuari u2	Èxit	Els missatges enviats arriben al receptor

#21 Desfer un match		
Escenari	**Resultat**	**Observacions**
L'usuari vol desfer un match ja actiu	Èxit	El sistema desfà el match i els usuaris ja no apareixen a la llista de matches de l'altre usuari

#22 Reportar usuari		
Escenari	**Resultat**	**Observacions**
L'usuari vol reportar un altre usuari amb qui ja té match o que li hagi enviat un missatge de match	Èxit	El sistema registra el report i desfà el match entre els usuaris

#23 Actualitzar les dades del nostre perfil		
Escenari	**Resultat**	**Observacions**
L'usuari canvia la seva bio i demana al sistema que actualitzi el seu perfil públic	Èxit	El sistema actualitza les dades públiques de l'usuari

#24 Actualitzar les fotos del nostre perfil		
Escenari	**Resultat**	**Observacions**
L'usuari afegeix, esborra o canvia les fotografies del seu perfil i demana al sistema que actualitzi les dades	Èxit	El sistema actualitza les fotografies de l'usuari

#25 Modificar la configuració del compte		
Escenari	**Resultat**	**Observacions**
L'usuari canvia la seva contrasenya i demana al sistema que actualitzi les dades	Èxit	El sistema actualitza les dades correctament
L'usuari canvia el seu gènere i demana al sistema que actualitzi les dades	Èxit	El sistema actualitza les dades correctament

#26 Modificar les preferències de l'algoritme de recomanacions		
Escenari	**Resultat**	**Observacions**
L'usuari actualitza el gènere de les recomanacions que li farà el sistema	Èxit	L'algoritme de recomanació s'adapta a les noves preferències

9.3. Estratègia de proves requisits no funcionals

A diferència de la validació dels requisits funcionals que estan directament lligats al codi, pels no funcionals ha calgut utilitzar altres tècniques, ja que alguns d'ells tractaven temes subjectius com pot ser la percepció. A continuació es detallen els tipus de proves que s'han realitzat:

- Comprovació manual: a partir de provar el comportament indicat al criteri de satisfacció amb diferents dispositius i repetint la prova si així és necessari (pels casos en què el criteri així ho indiqui), s'ha pogut determinar si un requisit ha estat aconseguit o no.
- Sessions de Think Aloud: el mètode "és un test en què se li demana al participant que faci servir un sistema mentre pensa contínuament en veu alta, verbalitzant els seus pensaments mentre es mou per la interfície d'usuari" [51]. Es van aprofitar les fires municipals per a organitzar una petita parada i realitzar els dos tests que requereixen aquesta metodologia.
- Comprovació visual: a diferència de la comprovació manual, per realitzar aquest tipus de test no cal fer cap interacció amb l'aplicació. Tests com la comprovació de la versió d'Android, l'encriptació de les contrasenyes a la base de dades o la presència de l'app a la tenda d'aplicacions requeririen aquest tipus de prova.

9.4. Validació dels requisits no funcionals

D'igual manera, per a validar els requisits no funcionals també es farà servir una mateixa taula per a tots els casos. En aquest cas el compliment del criteri de satisfacció serà el que determinarà si s'ha complert amb el requisit no funcional. Els atributs de la taula seran els següents:

- **Descripció**: descripció del requisit no funcional.
- **Criteri de satisfacció:** condició indispensable per a donar com a vàlid el compliment del requisit.
- **Resultat**: Resultat d'aplicar la prova determinada com a criteri de satisfacció.
- **Observacions**: comentaris o observacions que hagin sorgit de la realització de la prova.

Descripció	#1 El sistema ha d'estar disponible en més d'una llengua.	
Criteri de satisfacció	**Resultat**	**Observacions**
S'oferirà l'aplicació en català, castellà i anglès.	Èxit	L'idioma de l'aplicació es pot canviar a català, castellà i anglès des de la finestra per a gestionar el compte.

Descripció	#2 El sistema ha de recordar la llengua triada per l'usuari.	
Criteri de satisfacció	**Resultat**	**Observacions**
Es provarà que si es canvia la llengua i es tanca l'app, un cop es torni a obrir, la llengua en què es mostra és la que s'havia seleccionat. S'espera un encert del 100% dels casos.	Èxit	S'han realitzat 10 proves i totes han estat un èxit.

Descripció	#3 Ha de ser fàcil aprendre com fer servir el sistema després de completar un petit tutorial.	
Criteri de satisfacció	**Resultat**	**Observacions**
S'espera que el 80% dels usuaris que realitzin el test el superin satisfactòriament després de completar un petit tutorial.	Èxit	S'ha realitzat el test a 16 persones i 14 d'elles l'han superat sense requerir d'ajuda externa (87,5%). Al test es demanava realitzar algunes de les funcionalitats ofertes per l'app.

Descripció	#4 Les icones i les formes usades han de ser indicatius de la secció que representen.	
Criteri de satisfacció	**Resultat**	**Observacions**
S'espera que el 80% dels usuaris que realitzin el test sàpiguen orientar-se en l'aplicació a partir de les icones triades per a cada finestra.	Èxit	S'ha realitzat el test a 16 persones i 13 d'elles l'han superat sense requerir d'ajuda externa (81,25%). Al test es demanava localitzar determinades funcionalitats dins de l'app a partir de les seves icones.

Descripció	#5 L'algoritme de recomanació només pot mostrar usuaris d'esdeveniments que segueixes.	
Criteri de satisfacció	**Resultat**	**Observacions**
L'algoritme de recomanació només mostrarà usuaris d'esdeveniments que l'usuari segueix. S'espera que el 100% dels usuaris que realitzin el test només es trobin amb usuaris d'esdeveniments que segueixen.	Èxit	Dels 20 tests realitzats, tots han tingut un resultat positiu.

Descripció	#6 El sistema ha de funcionar en la majoria de dispositius Android.	
Criteri de satisfacció	**Resultat**	**Observacions**
El sistema farà servir una versió d'Android disponible en més del 70% de dispositius Android. El sistema farà servir una versió d'SDK 23, corresponent a la versió d'Android 6.0 Marshmallow, que funciona al 71% dels dispositius	Èxit	L'aplicació treballa amb el rang d'SDKs de la versió 23 a la 28, arribant per tant al 71% de dispositius disponibles

Descripció	#7 El sistema ha de ser distribuït des de la Play Store.	
Criteri de satisfacció	**Resultat**	**Observacions**
El sistema ha de passar els controls de la plataforma *Google Play* i ha d'estar disponible per a la seva descarrega sempre en l'última versió alliberada.	Èxit	L'aplicació es pot trobar a la Play Store buscant "PartyPal" o al link [52]

Descripció	#8 Les contrasenyes de la base de dades seran encriptades.	
Criteri de satisfacció	**Resultat**	**Observacions**
Totes les contrasenyes emmagatzemades a les bases de dades del sistema hauran de ser encriptades per tal d'augmentar la seguretat.	Èxit	Totes les contrasenyes guardades fan servir la funció de hash BCrypt

10. Resultats de la gestió del projecte

El següent capítol detalla els resultats que es desprenen de l'aplicació de les metodologies de treball triades, les desviacions respecte a la planificació temporal inicial i la planificació econòmica. També s'analitza la sostenibilitat del projecte.

10.1. Resultats a nivell de metodologia

10.1.1. Mètodes de treball

Encara que per a projectes amb un únic desenvolupador no se solen aplicar aquest tipus de metodologies, haver treballat seguint una metodologia agile ha dotat el procés d'una llibertat que no s'hauria pogut aconseguir amb un mètode tradicional: el desenvolupament incremental ha permès que, en moltes ocasions, memòria i codi es treballessin a l'hora, fins i tot encavalcant diferents fases. També ha permès refer o corregir apartats anteriors, com el de disseny, un cop ens hem adonat que hi havia punts que es podien millorar.

Un altre encert ha estat partir el temps en iteracions de dues setmanes. Aquesta duració ha permès treballar còmodament i amb marge suficient per a adaptar-me als altres requeriments de la universitat, com podrien ser projectes o exàmens d'altres assignatures. Aquests mateixos requeriments han obligat a treballar alguns caps de setmana per acabar els casos d'ús que estaven assignats per aquella iteració, però aquest era un escenari que ja es contemplava quan es va realitzar la planificació temporal. Que les iteracions hagin estat d'aquesta durada ha permès realitzar ràpides correccions per part de la tutora, evitant haver de realitzar grans canvis al final, com hauria succeït en una metodologia en cascada.

L'ús del product backlog també ha estat de gran ajuda. L'eina *Trello* ha permès visualitzar en tot moment en quines tasques s'havia de focalitzar l'atenció i quines es podrien deixar per més endavant. L'acció de parar-se a pensar quines funcionalitats volia que inclogués el projecte i granular-les en tasques factibles per les iteracions que s'havien dissenyat va ser una tasca més complexa del que a priori podia semblar, però de gran importància en el transcurs del projecte.

Per últim, encara que en el seu moment no es va pensar el mateix donada l'alta demanda del curs, el mòdul de GEP va permetre començar amb la documentació del projecte i em va obligar a passar de concepte o idea mental a una estructura escrita, ajudant-nos a identificar l'abast i a decidir quines funcionalitats es podien arribar a fer i quines no, aspecte que ara amb perspectiva s'agraeix molt.

10.1.2. Eines de seguiment

Les tres eines que s'han fet servir en aquest projecte han estat un encert total, ja que han complert la seva funció perfectament i han ajudat a organitzar i agilitzar el projecte de forma molt significativa.

La primera eina, *Trello*, ha estat vital per administrar el cicle de vida de les tasques per realitzar del projecte. Ha permès paral·lelitzar entre documentació i implementació i afegir ràpidament tasques noves que apareixien sobre la marxa. A la vegada, l'ús del bloc "If I have time" ha permès organitzar mentalment el projecte, ajudant al programador a ser realista amb el temps que restava fins a l'entrega final. El fet que *Trello* tingui versió web i versió app ha ajudat a fer que es puguin afegir notes en tot moment, evitant l'ús d'altres aplicacions secundàries.

La segona eina, *GitHub*, ha permès centralitzar tot el codi en un mateix lloc, simplificant l'alternança entre equips de treball de forma ràpida. La funció del controlador de versions també ha estat de gran ajuda, ja que la mateixa plataforma actualitzava automàticament els arxius que canviaven, sense crear-ne còpies. Per tant, sempre que es descarregava el codi, aquest es trobava en la seva última versió. El controlador de versions també ha permès desfer canvis al codi a versions anteriors, funcionalitat de gran utilitat quan s'ha volgut tornar a començar des d'un punt en concret. *GitHub*, en ser una plataforma tan estesa, ha fet possible que féssim servir la tercera eina, *CircleCI*, sense complicacions i amb un procés d'instal·lació estàndard i ràpid.

L'última eina de seguiment que s'ha utilitzat, connectada amb *GitHub*, ha estat *CirlcleCI*, plataforma encarregada de la integració contínua. De les tres eines ha estat, potser, la que ha tingut un impacte més directe en l'estalvi de temps de treball, ja que ha permès despreocupar-nos de la construcció, part del testing i del desplegament en producció només pujant el codi a *GitHub*. A la vegada, la funcionalitat d'avís en cas de fallada en la construcció del projecte ha fet que només haguem d'invertir energia en aquest procés quan sigui necessari, ajudant a mantenir la concentració en seguir desenvolupant-lo.

10.1.3. Mètode de validació

El mètode de validació dels tests ha estat altament efectiu en el transcurs del desenvolupament, perquè ha permès identificar i resoldre errors que apareixien al moment, i no en temps de producció o massa tard.

Per altra banda, les reunions amb la tutora, la Cristina Gómez, han estat vitals de cara a encarar els apartats que conformen aquesta memòria i per a corregir tant contingut com enfocament de les explicacions. Tant electrònicament com les reunions presencials realitzades, han estat de molta ajuda i s'han intentat aplicar totes les recomanacions que ha fet.

10.2. Resultats a nivell de la planificació

Atès que la planificació temporal va ser realitzada durant el mòdul de GEP, per al còmput d'hores que requeriria aquesta fase es van fer servir les emprades fins al moment, que foren 61. Aquesta era l'única dada real amb què es disposava llavors, la resta es van predir. És per aquest motiu que han sorgit desviacions respecte a la planificació inicial. La taula 9 mostra la diferència entre les hores calculades i les reals i a continuació es detallen els motius de les desviacions de cadascuna de les fases i els plans d'acció aplicats.

Tasca	Estimació del temps (en hores)	Hores reals	Diferència d'hores
1)GEP	61	61	0
2) Fase inicial	60	73.5	13.5
3) Primera iteració	60	44	-16
4) Segona iteració	60	66	6
5) Tercera iteració	60	57	-3
6) Quarta iteració	60	62	2

7)Cinquena iteració	60	64	4
8) Iteració final	61	40	-21
9) Preparació defensa	20	20	0
Total	502	487.5	-14.5

Taula 9: resum de les hores dedicades a les iteracions i anàlisis de les desviacions

Fase Inicial

La confecció dels documents va requerir més temps de l'esperat, perquè va ser necessari revisar material bibliogràfic i de cursos anteriors. En aquesta iteració estava planificada la confecció de tot l'apartat de disseny, però finalment es va decidir que una part d'aquest s'aniria elaborant de forma incremental, juntament amb l'apartat d'implementació. La motivació d'aquesta decisió va ser dotar de més llibertat i flexibilitat la presa de decisions durant el desenvolupament.

Primera iteració (01/04/2019 - 12/04/2019)

La familiarització amb els llenguatges i l'entorn de programació va ser més ràpid de l'esperat, ja que *SpringBoot* es basa en Java. La confecció del front end bàsic per a començar a treballar va requerir el temps calculat.

Segona iteració (15/04/19 - 26/04/19)

En aquesta iteració no solament es va sobrepassar l'estimació de les 60 hores, sinó que a més no es va arribar a poder implementar tots els casos d'ús programats. Durant la planificació inicial no es va tenir en compte l'esforç que requeriria configurar l'entorn de treball. Afegit a això, durant l'inici de la iteració es va comprovar que a causa de restriccions de permisos de la xarxa de la universitat, no es podia accedir a altres dispositius a partir de la seva IP, aspecte imprescindible per comunicar el servidor i l'APP si es volia treballar en local. La solució fou migrar directament a un servidor extern, tasca que correspondria a la fase final de la planificació. Aquest imprevist va fer que no donés temps a realitzar el cas d'ús corresponent al sistema d'esdeveniments, que es va deixar per a la tercera iteració.

Tercera iteració (29/04/19 - 10/05/19)

En la tercera iteració es van aconseguir realitzar tots els casos d'ús programats més l'afegit de la iteració anterior. Es va fer en un temps una mica inferior a l'estimat, fet que ha permès equilibrar el balanç d'hores extres realitzades.

Quarta iteració (13/05/19 - 24/05/19)

En la quarta iteració es van poder finalitzar tots els casos d'ús programats, però es va requerir una mica més de temps de l'estimat, perquè *Firebase*, una de les bases de dades que s'han fet servir en aquest projecte, té un comportament asíncron, aspecte incompatible amb la visió síncrona de la nostra API. Finalment es va poder trobar una solució, però va requerir invertir un nombre d'hores considerable en la recerca d'aquesta.

Cinquena iteració (27/05/19 - 07/06/19)

En la cinquena iteració es va finalitzar la tasca de gestió dels matchs en el temps estimat, però es va requerir més temps per al desenvolupament del sistema de xat. Aquest endarreriment va ser degut a la confecció del front end de la funcionalitat i a una posterior millora de la gestió que es feia dels xats al sistema *Firebase*, per tal de millorar l'escalabilitat de l'aplicació. Tot i això, es va optar per acabar aquest cas d'ús en aquesta iteració, encara que requerís una mica més d'esforç.

Fase final

La fase final es va allargar del 10 al 25 de juny. Atès que a la segona iteració ja em vaig veure obligat a migrar al servidor de producció, i aquesta tasca estava programada per a la fase final, les hores que han calgut dedicar han estat menys de les programades. En aquesta fase s'han acabat de polir detalls de disseny de l'app mòbil i del treball escrit, acabant els apartats de validació i proves, l'actual apartat i les conclusions.

Preparació defensa

La preparació de la defensa és l'última fase del projecte i es va programar per començar tan bon punt es tanqués la memòria. Com que encara no s'ha tancat, s'estima que requerirà el temps que es va estipular durant la fase inicial.

Tal com es veu a la taula 9, la dedicació real ha estat de 487.5 hores amb una desviació total en nombre d'hores respecte a la planificació inicial de -14.5, és a dir menys de les estimades en un començament. Tot i que la data de finalització ha estat gairebé la mateixa que es va planejar, aquesta petita reducció en la càrrega en hores ha estat deguda a la velocitat d'aprenentatge, l'adaptació a les tecnologies aplicades i a treballar directament en el servidor de producció, estalviant-nos la migració final.

Aquesta desviació equival a un 2,98% respecte a la planificació original, fet que es considera molt positiu tenint en compte la poca experiència en planificació de l'autor del treball.

10.3. Resultats a nivell de gestió econòmica

A partir de les desviacions temporals estudiades al punt anterior, cal determinar si el possible increment en el cost del projecte encara es troba dins del pressupost que es va plantejar o aquest s'ha quedat curt.

En el capítol 4 es mostrava la taula 7, que resumia el pressupost final predit del projecte. Per comoditat es torna a mostrar aquesta taula a continuació.

Activitat	Import
Costos directes	9.610,23€
Costos indirectes	513,80€
Imprevistos	1.518,60€
Contingències	380,00€
Total	**12.022,63€**

Taula 10: pressupost final del projecte

Com que el pressupost es va haver de realitzar a priori i es van haver de predir la implicació que cada un dels rols dedicaria en cada iteració, aquests costos també han sofert desviacions. Per tal de veure si aquestes estan cobertes per la partida de contingències, s'han tornat a calcular els costos directes amb les dades definitives.

Fase (dedicació estimada)	Rol (dedicació en hores)				Total
	Cap de projecte	Analista	Dissenyador	Programador i tester	
GEP (61.1 h)	61				1,732.95€
Fase inicial (60 h)	13.5	45	15		1,470.17€
Primera iteració (60 h)	5		4	35	773.30€
Segona iteració (60 h)	10		5	51	1,192.61€
Tercera iteració (60 h)	2		2	53	957.39€
Quarta iteració (60 h)	5		4	53	1,069.89€
Cinquena iteració (60 h)	2		2	60	1,072.73€
Iteració final (61 h)	22		7	11	901.70€
Preparació defensa (20 h)	20				568.18€
Total	140.5	45	39	263	**9,738.92€**

Taula 11: cost per rol de cadascuna de les fases del projecte actualitzada

La taula 10 mostra com els costos directes s'han incrementat en només 129 €. Aquest import encara entra dins de la partida de contingències, que es va fixar en 380 €.

Com a imatge global, a part dels 250 € que encara resten de la partida de contingències, tampoc ha calgut usar els 1518 € de la d'imprevistos. Encara que aquesta última dada no és significativa, ja que no es pot predir quan succeirà algun fet inesperat i potser en un altre projecte aquesta partida es queda fins i tot curta, sí que ho és el fet de no haver necessitat la totalitat de la partida de contingències, perquè indica una bona planificació inicial.

10.4. Sostenibilitat

Tots els TFGs presentats a la FIB han d'incloure un apartat de reflexió sobre els impactes sostenibles del treball. Es treballaran 3 dimensions: l'econòmica, l'ambiental i la social.

10.4.1. Sostenibilitat econòmica

Quan s'intenta realitzar un projecte econòmicament sostenible, és important no treballar sobre la marxa i realitzar una avaluació de costos completa, tant de recursos humans com materials, com és el cas d'aquest projecte. A més, s'ha tingut en compte possibles ajustaments i

actualitzacions del ritme de treball i s'ha preparat una partida per a contingències i per a imprevistos de forma realista.

En ser un projecte educatiu, la principal finalitat d'aquest és plasmar els coneixements adquirits durant el transcurs del grau i afegir-ne de nous, per tant, resulta evident que es podria realitzar en menys temps si ja es tingués experiència en les tecnologies que s'estan usant. A la vegada, també es podria realitzar en menys temps si l'equip fos més gran, però això augmentaria el cost final.

El temps dedicat a cada tasca, a priori, sembla l'adequat: s'ha assignat un marge de temps gran per a familiaritzar al desenvolupador amb totes les eines i llenguatges que necessitarà durant el transcurs del projecte i s'ha intentat evitar "reinventar la roda" tant com s'ha pogut, fent ús d'APIS de companyies externes que ja ofereixen un bon servei, estalviant el temps que s'hauria d'haver dedicat a desenvolupar-lo nosaltres.

Un altre avantatge de ser un projecte educatiu és que el pressupost final és altament competitiu. Encara que el projecte sigui petit, s'ha aconseguit encabir tots els costos en un pressupost inicial de 12.022€. En el pressupost final s'han necessitat 129 € més per la partida dels costos directes, però s'han mitigat amb el coixí de contingències. S'estima per tant que l'impacte econòmic en producció té una nota de **9/10**.

Ara mateix, una solució parcial a aquest problema (si fos completa, el projecte no tindria sentit), l'ofereixen grans empreses com *Facebook, Tinder, Twitter*… companyies amb plantilles, pressuposts i petjada econòmica i ambiental molt més gran que la que s'estima d'aquest projecte. En el nostre cas, un cop el producte es trobi disponible al públic, si no es volgués afegir res més i només calgués mantenir-lo, l'únic cost que tindria és el del servidor, que és relativament baix (en el nostre cas 5€/mes). Si la demanda puja i calgués un altre servidor, s'estudiaria on posicionar-lo per distribuir la càrrega que haurien de suportar i així treure'l-hi més profit. S'estima que l'impacte econòmic en vida útil serà també de **9/10**.

Un risc real del projecte podria ser que una aplicació de la competència que comptés amb més quota de mercat decidís incorporar el nostre sistema com una de les seves funcionalitats. En aquest cas és clar que el peix gran es menjaria el petit. Així i tot, sempre es podria adaptar el mercat potencial a qui va dirigida la nostra aplicació i adaptar-se a la nova situació. S'estima que l'impacte econòmic davant d'aquest risc és alt i la solució quedaria lluny de ser suficient. Es dona a aquest punt un **3/10**.

10.4.2. Sostenibilitat ambiental

Potser la més difícil de quantificar en un projecte software, la sostenibilitat ambiental també és tinguda en compte. El recurs principal que es fa servir és l'electricitat. No cal cap hardware, no es produeix CO_2 durant l'elaboració del treball, es pot realitzar amb equips ja amortitzats… L'únic recurs que es necessita de l'usuari és emmagatzematge al dispositiu mòbil i bateria, i en els dos casos el cost és mínim.

Un altre aspecte on la sostenibilitat ambiental s'analitza és en l'elecció de servidors per a allotjar el nostre servei. A l'hora de realitzar la tria per veure quin és el més adequat, no només s'ha tingut en compte un criteri econòmic, sinó també ambiental: s'ha triat la màquina que complia les necessitats del projecte, sense aspirar a una més potent i per tant, amb més impacte.

Per últim, s'ha intentat reutilitzar tants recursos com es pot: no cal hardware nou i pel que fa a la programació, es fan servir APIs externes, evitant el temps que es trigaria a aconseguir oferir el mateix servei nosaltres mateixos, estalviant aquesta energia. Per tant, s'estima que la sostenibilitat ambiental en el procés de producció del projecte ha estat **10/10**.

Durant la vida útil del projecte, el sistema requerirà un mínim ús de la bateria per part de l'usuari i de l'energia que gasti el servidor que es té contractat. Atès que el servidor no és propi i és subcontractat i que el preu que es paga és reduït, aquest fet fa pensar que l'empresa proveïdora intentarà optimitzar al màxim l'ús de recursos per maximitzar el guany. S'estima que la sostenibilitat ambiental durant la vida útil del projecte serà de **9/10**.

Per acabar, un risc identificable seria un augment en l'ús de la bateria si no s'actualitzés l'app per adaptar-se a les noves versions d'Android. Així i tot, ja que la política de la plataforma és donar suport a com més aplicacions millor, s'estima que l'impacte serà llunyà i molt mínim. Per aquest apartat es posa una nota de **8/10**.

10.4.3. Sostenibilitat social

Personalment, aquest projecte m'ha ajudat a obrir encara més els ulls a una revolució social que està succeint actualment. Ensenyant-li l'aplicació ja "acabada" a unes amigues, en l'apartat de configuració del compte es van adonar que deia "seleccioni el seu sexe" just a sobre d'un desplegable amb les opcions *Home, Dona, Altre i Prefereixo no contestar*. La seva primera reacció fou dir-me que usar la paraula "sexe" era molt limitant, i que m'aconsellaven usar "gènere". Per descomptat vaig fer el canvi la mateixa tarda. Ja que estàvem, els hi vaig demanar consell respecte a les opcions que hauria de mostrar en el desplegable de "Quin gènere haurien de tenir els usuaris de les recomanacions?". És en aquest punt on em vaig adonar que la pregunta no era tan simple com hauria imaginat, ja que no he trobat una llista oficial de tots els gèneres o orientacions sexuals que existeixen actualment, i cada cert temps n'apareixen de noves. La solució a què s'ha arribat, i sent conscient que per molta gent serà insuficient, ha estat oferir les opcions Home, Dona i Tothom. Encara que aquest detall pugui semblar una ximpleria a ulls del lector, per a mi com a desenvolupador ha estat una decisió molt meditada i demano disculpes si no ha estat la millor opció. Així i tot, per l'aprenentatge realitzat i per la intenció de correcció, en aquest apartat estimo que l'impacte social en la producció del projecte ha estat **6/10**.

Tot i que el projecte entra en la categoria d'aplicacions d'oci i sempre existirà l'etern debat si l'oci és realment necessari o no, la proposta darrere d'aquest treball és innovadora i pot cobrir una necessitat que potser la gent encara no sap que té. Es fa servir un mètode que ja funciona, i per tant la gent ja coneix, que és el de les aplicacions matchmaking, i se li apliquen un seguit de millores focalitzades en un sector del mercat en particular.

La finalitat del projecte no és canviar la vida de cap dels usuaris que facin servir el producte, però sí fer-la més fàcil i entretinguda. La socialització ha estat sempre una de les pedres angulars de la nostra espècie, i la possibilitat de poder filtrar en un tema determinat aquesta socialització segur que és benvingut per molta gent. S'estima que la sostenibilitat social durant la vida útil del projecte serà de **9/10.**

Hi ha dos riscos importants que podrien ser perjudicials pels usuaris de l'aplicació. El primer és la suplantació d'identitat, on algú es faria passar per una altra persona mostrant les seves fotografies i/o nom. L'altre risc deriva del sistema de xat. Dins d'aquest no tenim control dels missatges que es comparteixen i de les situacions que es puguin viure. La solució per tots dos

riscos és el sistema de reports, on es pot denunciar un comportament inadequat d'un usuari dins l'aplicació. S'estima que es dona una resposta coherent als riscos que comporta la sostenibilitat social i se li dona a aquest punt un **8/10**.

	PPP	Vida Útil	Riscos	Total
Econòmica	9	9	3	21/30
Ambiental	10	9	8	27/30
Social	6	9	8	23/25
Total	25/30	27/30	19/30	71/90

Taula 12: Matriu de sostenibilitat del TFG

La taula 12 mostra la matriu de sostenibilitat d'aquest projecte i les seves puntuacions en cada apartat. Encara que millorable, es determina que aquest projecte ha tingut cura de totes les dimensions i que és sostenible.

11. Conclusions

11.1. Reflexions finals

Un cop arribats a aquest punt del projecte, només queda fer retrospectiva de tot el treball realitzat fins ara.

Tal com indica el resum inicial de la memòria, “El sistema que s’ha de desenvolupar busca oferir un punt d’unió en forma de xat entre persones que assistiran a un mateix esdeveniment abans de la seva celebració. L’aplicació oferirà l’opció de seguir els esdeveniments en els quals puguis estar interessat i recomanarà usuaris que també hi assistiran; permet, per tant, la socialització selectiva: hi ha un tema comú.”. L’objectiu original d’aquest projecte fou viure el procés de creació tècnic i es va determinar que una plataforma online seria la millor opció, donada l’experiència que acumulem la generació millennial. Aquesta plataforma havia de possibilitar a l’usuari filtrar entre els esdeveniments en què podia estar interessat i crear una mena de “joc” per a conèixer gent nova.

La font d'inspiració d'aquesta idea fou l'experiència pròpia d'anar a un festival de tres dies sol i adonar-me'n que socialitzar era molt més fàcil del que m'hauria imaginat, perquè d'entrada, sabíem les dues persones que ja teníem un tema de conversa en comú, que era la raó per la qual estàvem allà.

A força de donar-li voltes vaig veure que ben orientat, podria tenir futur com a aplicació i ho vaig apuntar a aquella llista que tots portem al mòbil i que cada cop es fa més llarga de "projectes per fer". La cosa va quedar allà fins que em va tocar decidir tema pel meu treball de fi de grau i vaig pensar que potser seria una bona ocasió. Li vaig presentar a la Cristina i encara que no veia molt clar en l'àmbit tècnic com ho faria, em va donar suport des del primer moment.

L’odissea de transformar un concepte mental en una realitat ha estat una aventura meravellosa. Quan parlava amb alumnes de cursos anteriors em comentaven que és realment en la confecció del TFG quan t’adones de tot el que has anat aprenent durant la carrera i ho vas connectant, i realment això és el que he fet. Aquest projecte toca àmbits totalment diferents dins de l’enginyeria del software i potser una mica més enllà: el component social. Durant el transcurs del treball, l’objectiu sempre ha estat crear un producte final útil, que d’alguna manera cobrís una necessitat de l’usuari, que segurament ni sabia que tenia. M’he adonat que la finalitat real dels informàtics és la de crear serveis que puguin, en la mesura del possible, millorar la qualitat de vida dels usuaris que els facin servir.

En finalitzar el projecte, s’han complert satisfactòriament tots els objectius inicials que es van establir: s’ha desenvolupat un sistema integral, tant front com back end, s’han usat les tecnologies punteres del mercat, s’han aplicat sistemes d’integració contínua, s’ha integrat el sistema amb serveis externs... a part de tot el procés manual d’analitzar i especificar l’abast, els requisits o el disseny del projecte, entre d’altres.

Afegit a la satisfacció de les funcionalitats implementades, està el coneixement adquirit. La gran majoria d’aquest ve de les parts que no han funcionat a la primera, més que les que sí. Hores i hores davant d’un error per acabar fins i tot simpatitzant amb ell.

De nou, un cop en aquesta posició ja a punt de finalitzar, faig autocrítica i reconec que segurament hi haurà determinats apartats, funcionalitats, patrons de disseny o enfocaments del

codi que potser no són l'aproximació o l'opció més òptima, però com a imatge global, expresso la meva satisfacció amb el resultat final. Aquest projecte ha estat la conclusió d'haver anat recollint durant aquests quatre anys peces d'un puzle que en el seu moment no podia imaginar com encaixarien, però que han format un resultat final del qual estic molt orgullós.

Com a última dada, un mes abans del tancament d'aquesta memòria i sense avís previ, *Tinder*, l'empresa matchmaking més gran del món, va afegir la funcionalitat "Festival Mode", que encara que no permet filtrar només per esdeveniments com fem nosaltres, sí que convida a l'usuari a decidir si una recomanació és del seu grat o no depenent dels festivals als quals hi vagi. És obvi que no tenim cap possibilitat de lluitar contra *Tinder*, és un peix massa gran, però aquest moviment indica que la nostra idea no era tant "boja" i que estava ben encaminada.

Per últim, m'agradaria donar les gràcies a tota la gent que, de forma desinteressada, han participat en la confecció d'aquest projecte, independentment de com.

M'agradaria agrair també la col·laboració i dedicació de la meva tutora, la Cristina Gómez Seoane, que sense deixar de confiar en mi, m'ha guiat en tot aquest procés.

11.2. Limitacions i dificultats

Encara que el desenvolupament d'aquest projecte ha estat fluid, a continuació es detallen algunes de les dificultats i limitacions significatives que han alterat en un grau o un altre el desenvolupament del sistema.

- Treballar en la xarxa universitària: ja he explicat al capítol anterior, que quan es va començar a programar l'aplicació, la intenció era treballar en local replicant l'estructura que després es crearia al servidor. Per a la nostra sorpresa, encara que per raons de seguretat té sentit, la xarxa de la universitat, que seria en la que treballaríem la major part del temps, no permetia realitzar accessos dispositiu a dispositiu via IP. Aquesta tècnica la volíem fer servir per connectar el telèfon mòbil de desenvolupament directament amb l'ordinador, però no va funcionar. Ara que ja gairebé s'ha acabat el projecte, aquesta pedra al camí ens va permetre treballar des de primer moment en el servidor final. És cert que es van dedicar esforços per configurar-lo correctament i per incorporar el sistema d'integració contínua perquè ens ajudés amb les tasques de desplegament, però la feina que ens ha estalviat en el transcurs de totes les iteracions ha estat immensa.
  A més, ha permès treballar des de qualsevol dispositiu, sense que calgués introduir manualment l'adreça IP amb el que estàvem treballant actualment. Un altre avantatge que ha aportat ha estat que arribat el final del projecte, el sistema ja estava funcionant sense haver de patir per realitzar la migració, instal·lació de llibreries, compatibilitat de versions...

- *Firebase*: des d'un primer moment s'era conscient que per aquest treball es farien servir dues bases de dades: una local al servidor relacional per a guardar les dades "estàtiques" (registre d'usuaris, esdeveniments, matchs, peticions de match...) i una remota noSQL per a les peticions "dinàmiques" (sessió dels usuaris, xats actius, missatges...). Per a la base de dades noSQL es va optar per *Firebase*, desenvolupada per *Google*, donada l'acceptació dins la comunitat de desenvolupadors i perquè oferia les funcionalitats requerides. En un punt del desenvolupament de l'algoritme de recomanacions es va pensar migrar una part de la base de dades local a *Firebase*, però ens vam topar amb una dificultat: *Firebase* és un sistema asíncron, mentre tot el nostre sistema estava pensat per ser síncron. És a dir, quan l'app mòbil realitza una crida al nostre servidor

espera resposta en un temps definit; si aquesta no arriba, es considera que la petició ha fallat i la torna a enviar. En *Firebase* es realitza una consulta i es programa un listener perquè en un moment del futur no determinat, quan arribi la resposta, es faci el tractament d'aquesta. Aquests dos plantejaments no funcionen bé junts.
Davant d'aquest problema es van plantejar dues solucions: migrar tot el sistema a un plantejament asíncron o migrar la gestió de les funcionalitats que requerissin de *Firebase* a l'app. Es va triar la segona ja que realment l'algoritme de recomanacions es podia gestionar amb la base de dades local i l'única funcionalitat que realment requeria de *Firebase* era el xat, que des d'un començament ja s'havia dissenyat que la seva lògica aniria a l'app per qüestions d'estalvi de comunicació contínua entre app i servidor.

- Play Store: com que el sistema tracta dades de caràcter confidencial i més amb l'aplicació de la nova llei de protecció de dades, *Google* requereix que s'especifiqui quin serà el tractament que es farà d'aquestes i obliga a proporcionar una còpia escrita tant dins l'aplicació com de forma externa (sigui una pàgina web). Atès que no es tenia en compte aquesta dificultat i la falta de coneixements jurídics per escriure uns termes i condicions d'ús de manera autònoma, s'ha optat per copiar i enganxar els d'alguna aplicació important realitzant mínims canvis. L'autor és conscient que aquesta no pot ser una solució definitiva perquè cap document jurídic existent s'acabarà adaptant 100% al nostre sistema, però deixa per al treball futur la seva confecció. En el moment d'escriure aquestes línies, *Google* està tornant a revisar els nous termes que s'han penjat amb l'aplicació.

11.3. Integració de coneixements

Per tal de demostrar la transversalitat del projecte i la integració del coneixement après de diferents disciplines, es procedeix a llistar un seguit d'assignatures del grau el temari de les quals ha estat aplicat directament en la confecció d'aquest treball.

Estructura de Dades i algorismes (EDA):

Dins del projecte es fan servir tant estructures explicades a classe (maps, llistes...) com algoritmes de cerca. Cert és que l'algoritme de recomanació no és cap dels que recull l'assignatura, però s'ha tingut en compte el cost de les diferents alternatives per a fer-lo el més eficient possible.

Bases de Dades (BD):

El sistema desenvolupat fa servir dues bases de dades per a guardar tota la informació que es genera: una *PostgreSQL* i una *Firebase* (NoSQL). Tota la gestió de *Postgre* es fa a partir de les comandes i l'enfocament explicat a l'assignatura.

Interacció i Disseny d'Interfícies (IDI):

El sistema el compon un servidor i una app Android. Per a aquesta s'han aplicat els principis de disseny d'aplicacions explicats a l'assignatura.

Xarxes de Computadors (XC):

La comunicació entre l'app i el servidor es fa a partir del protocol TCP/IP, on intervenen les adreces IP tant del servidor com dels dispositius. A més, ha estat important fer el tractament de

la latència de la xarxa per a distingir quan una dada encara es troba en trànsit o quan no hi ha hagut resposta.

Projectes de Programació (PROP):

Tant l'app Android com el servidor estan implementats en Java, llenguatge explicat a l'assignatura. A més, s'ha aplicat orientació a objectes en els dos casos.

Arquitectura del Software (AS):

L'arquitectura del Software és vital per a dissenyar tot el sistema, ja sigui en el marc teòric com és l'elaboració dels apartats de disseny o d'implementació dins de la memòria, com en el marc pràctic a l'hora de prendre decisions en directe a mesura que es van programant els diferents components.

Aplicacions i Serveis Web (ASW):

Durant el transcurs de l'assignatura es van provar algunes de les tecnologies principals per al desenvolupament d'aplicacions i serveis web moderns, experiència que s'ha aprofitat a l'hora de decantar-se per un framework que oferís les característiques desitjades. També s'ha seguit el consell de l'assignatura a l'hora de fer una aproximació REST a l'API.

Disseny de Bases de Dades (DBD):

Si de l'assignatura de Bases de Dades es va aprendre com funcionava una base de dades relacional i les comandes principals per a fer-la servir, de l'assignatura de Disseny de Bases de Dades s'han aplicat molts dels coneixements apresos per a triar l'estructura de les taules i la freqüència d'accessos.

Enginyeria de Requisits (ER):

Per a la realització dels apartats d'Anàlisi de requisits i el d'Especificació ha estat vital el coneixement que es va adquirir durant la realització de l'assignatura d'Enginyeria de Requisits. S'han aplicat les tècniques i les metodologies tal com es van explicar.

Gestió de Projectes de Software (GPS):

Tot i que l'assignatura de Gestió de Projectes de Software se centrava en l'organització del treball en equip, les eines que es van ensenyar han estat utilitzades en aquest treball donada l'ajuda en la simplificació de tasques que ofereixen. Eines com *Trello* i el concepte d'integració contínua van ser ensenyats en aquesta assignatura.

Projecte d'Enginyeria del Software (PES):

L'assignatura de Projecte d'Enginyeria del Software va ser un preludi pel que seria la realització de l'actual projecte. En aquesta es van aplicar tots els coneixents que s'havien anat adquirint durant el grau i l'especialitat i es posaven en pràctica en un treball en grup. El projecte actual està fortament influenciat per les tecnologies i metodologies que es van seguir a PES.

11.4. Justificació de les competències

El següent apartat detalla com s'han complert les competències tècniques associades a aquest projecte especificades durant la matriculació del TFG.

CES1.1: Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics. [En profunditat]

El propòsit central del projecte i el fil conductor d'aquest ha estat el desenvolupament d'un sistema complex, el manteniment del seu codi a mesura que avançava el procés i l'avaluació de les diferents alternatives aplicables a cadascuna de les funcionalitats que s'han anat implementant. El desenvolupament del sistema no s'ha limitat a una aplicació auto continguda, sinó que s'han programat dos components, una app i un servidor, fent ús de bases de dades tant locals com externes, a més de sistemes externs de tercers. El projecte, per tant, s'ha desenvolupat amb èxit, completant totes les etapes definides en el transcurs del grau, pel que es considera una competència assolida i en profunditat.

CES1.2: Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles. [Bastant]

Tot i que no va ser escollida com una competència en profunditat, se n'ha fet un tractament gairebé igual. D'entrada, ha calgut treballar la integració entre l'app mòbil i el servidor, comunicació vital perquè el sistema funcionés. Per altra banda, també ha estat necessària la integració amb els serveis externs i el tractament de les dades conseqüent. S'ha fet servir el sistema d'autenticació de *Google*, el sistema de cerca d'esdeveniments de *Ticketmaster* i la base de dades en temps real *Firebase*. La competència es considera assolida.

CES1.3: Identificar, avaluar i gestionar els riscos potencials associats a la construcció de software que es poguessin presentar. [Una mica]

Malgrat aquesta competència es va triar en un grau d'implicació baix, el procés d'anàlisi i especificació dels requisits va permetre detectar i corregir riscos potencials en una etapa prèvia per començar a desenvolupar el projecte. D'igual manera, en haver fet servir una metodologia incremental, quan apareixia algun risc no identificat, es podia gestionar la seva solució en un període de temps curt.

CES1.4: Desenvolupar, mantenir i avaluar serveis i aplicacions distribuïdes amb suport de xarxa. [Una mica]

Igual que en l'anterior, encara que es va marcar com "una mica", l'impacte que aquesta competència ha tingut en el projecte ha estat alt. El sistema l'integren dos components: un front end, disponible al dispositiu del client que es descarregui l'app, i un back end, funcionant al nostre servidor de producció situat a Frankfurt. Perquè l'aplicació funcioni ha calgut especificar els temps d'espera i les polítiques de re intent entre petició i petició, així com gestionar la cua entre peticions. També s'han fet servir sistemes externs amb un comportament asíncron, fet que obligava al sistema a realitzar altres operacions mentre esperava la resposta.

CES1.5: Especificar, dissenyar, implementar i avaluar bases de dades. [Una mica]

En aquest projecte, encara que no s'ha centrat plenament en la gestió de les bases de dades, aquestes han tingut un paper molt important donada la necessitat de preservar la informació

introduïda al sistema. El nostre sistema ha treballat amb dues bases de dades diferents: la primera, allotjada al mateix servidor que el programa, relacional i basada en SQL. La segona, *Firebase*, ha estat una base de dades noSQL. Per tant, encara que es va marcar com "una mica", la gestió que s'ha fet d'aquesta competència es considera satisfactòria.

CES1.7: Controlar la qualitat i dissenyar proves en la producció de software. [Bastant]

La qualitat i presència de les proves ha estat vital en el correcte desenvolupament del projecte i es mostra en la dedicació de tot un capítol (9. Validació i proves) en la memòria. S'han provat tant els requisits funcionals com els no funcionals, aplicant tests automatitzats i manuals. Dins dels tests automatitzats, s'han programat un conjunt de 69 tests tant unitaris com d'integració i aquests s'han executat cada cop que el sistema d'integració contínua ha construït el projecte per desplegar-lo al servidor de producció. A més, ha estat una pràctica comuna en els casos manuals provar més d'un escenari possible per aplicar triangulació en les proves. Per acabar, dins dels requisits no funcionals s'han aplicat tres estratègies de comprovació diferents. Es considera que s'ha assolit satisfactòriament aquesta competència.

CES2.1: Definir i gestionar els requisits d'un sistema software. [Bastant].

Dins del projecte s'han estudiat tant els requisits funcionals com els no funcionals que guiarien el posterior desenvolupament. Als capítols 5. Anàlisi de requisits i 6. Especificació es realitza un estudi detallat dels diferents requisits del sistema, organitzats per grups d'usuaris. Els requisits no funcionals es relacionen amb els de Volere i es proporciona un criteri de satisfacció. Pels requisits funcionals es determina l'escenari principal d'èxit i escenaris secundaris. L'extensa documentació present a la memòria i la dedicació que s'ha posat en aquesta competència fa que es consideri com a satisfeta.

11.5. Treball futur

Per acabar, encara que personalment el projecte en general el considero un èxit, hi ha un petit llistat de punts a implementar/millorar si s'hagués disposat de més temps. Es detallen a continuació:

- Iniciar sessió amb diversos serveis: actualment es permet iniciar sessió amb un compte de *Google*, en un futur es podrien incorporar serveis com *Twitter* o *Facebook*, entre d'altres.

- Millorar la interfície d'usuari: encara que amb la constant evolució de les modes en disseny gràfic arribar a una interfície definitiva és gairebé una utopia, en un futur es podria treballar en aquesta, apropant-la encara més als patrons de disseny més actuals. El disseny MVC del sistema ha de permetre que es pugui canviar tant com es vulgui de la part visual de l'aplicació sense haver de realitzar gairebé canvis a la lògica d'aquesta.

- Web d'administració: actualment l'administració de tot el contingut de l'aplicació es realitza a partir de la interfície pròpia de *Firebase* o d'un gestor d'SQL, en el nostre cas, *PostgreSQL*. En un futur podria ser d'utilitat implementar una web d'administració que unifiqués aquestes dues gestions i altres funcionalitats.

- Termes i condicions d'ús definitius: Ja comentat anteriorment, els termes i condicions d'ús que s'han fet servir tant en l'aplicació com en la play Store per a poder penjar-la

han estat un “còpia-enganxa” d'un altra aplicació ja existent al mercat realitzant mínims canvis. En un futur és necessari crear-ne uns personalitzats i adaptats al mateix projecte.

- Algoritme de recomanació: com que la finalitat principal d’aquest projecte no ha estat centrar-se en l’algoritme de recomanació sinó en la imatge global del sistema en si, aquest és un algoritme senzill. En un futur, es podria intentar explotar les dades de les que disposa el sistema i fer que l’algorisme pogués realitzar recomanacions molt més acurades.

- Encriptació del xat: encara que el sistema sí que encripta les contrasenyes, en un futur i seguint la tendència actual a encriptar com més parts del sistema millor, es podrien encriptar tots els missatges que es compartissin dins l’aplicació. Caldria estudiar quin impacte en el rendiment tindria aquesta acció.

- Proveïdors de serveis: per aquest projecte s'ha fet servir *Ticketmaster* per la quantitat d'esdeveniments que oferia la seva base de dades respecte als competidors, però aquests també operen en altres sectors, potser més locals, que també estaria bé incorporar a l'aplicació. D'aquesta manera s'estaria ampliant el mercat potencial al qual podem arribar i reforçant la imatge de la marca enfront de la competència.

12. Bibliografia

[1] Robertson, S.; Robertson, J,. Scoping the Business Problem. Robertson, S.; Robertson, J,. *Mastering the requirements process: getting requirements right*. Tercera edició. Upper Saddle River : Addison-Wesley, cop. 2013. ISBN 9780321815743 (cart.)

[2] Amplify Media. *PartyWith - Locals & Travelers*[en línia]. Amsterdam: Amplify Media, 2018. [Consulta: 20 febrer 2019]. Disponible a: <https://www.partywith.co/>

[3] Matt Schrage. *Kickback - Find Local Events & Things Going On Near Me - Discover Nearby Bars, Parties & Night Clubs*[en línia]. Matt Schrage, 2015. [Consulta: 20 febrer 2019]. Disponible a: <https://itunes.apple.com/us/app/kickback-find-local-events-things-going-on-near-me/id961537928?mt=8>

[4] Expreem. *Expreem - Meet New People Nearby, Find Friends and Local Activities*[en línia]. Expreem, 2015. [Consulta: 20 febrer 2019]. Disponible a: <https://itunes.apple.com/us/app/expreem-meet-new-people-nearby-find-friends-local-activities/id718334302>

[5] Tinder Inc. *Tinder*[en línia]. Tinder Inc., 2012. [Consulta: 20 febrer 2019]. Disponible a: <https://tinder.com/?lang=es-ES>

[6] Facebook Inc. *Facebook*[en línia]. Facebook Inc., 2004. [Consulta: 20 febrer 2019]. Disponible a: <https://www.facebook.com/>

[7] Badoo Software Ltd. *Badoo - Conoce gente nueva*[en línia]. Badoo Software Ltd, 2006. [Consulta: 20 febrer 2019]. Disponible a: <https://badoo.com/>

[8] Ticketmaster. *Ticketmaster*[en línia]. 1976. [Consulta: 20 febrer 2019]. Disponible a: <https://www.ticketmaster.com/>

[9] Eventbrite. *Eventbrite*[en línia]. 2006. [Consulta: 20 febrer 2019]. Disponible a: <https://www.eventbrite.es/>

[10] Trello Inc. *Trello*[en línia]. 2011. [Consulta: 20 febrer 2019]. Disponible a: <https://trello.com>

[11] Microsoft i GitHub Inc. *GitHub*[en línia]. 2008. [Consulta: 20 febrer 2019]. Disponible a: <https://github.com>

[12] CircleCI. *CircleCI*[en línia]. 2011. [Consulta: 20 febrer 2019]. Disponible a: <https://circleci.com/>

[13] Departament d'Organització d'Empreses. *Gestión del tiempo* [en línia]. Barcelona: Facultat d'Informàtica de Barcelona (FIB). [Consulta: 4 març 2019]. Disponible a: <https://atenea.upc.edu/pluginfile.php/2669389/mod_folder/content/0/M%C3%B2dul%202.3.1%20-%20Gesti%C3%B3n%20del%20tiempo.pdf?forcedownload=1>

[14] Dan Radigan. *The Product Backlog: Your Ultimate To-Do List* [en línia]. Atlassian, 2016. [Consulta: 3 març 2019]. Disponible a: <https://es.atlassian.com/agile/backlogs>

[15] Gantt.com. *What is a Gantt Chart? Gantt Chart Software, Information, and History* [en línia]. Gantt.com. [Consulta: 8 març 2019]. Disponible a: <https://www.gantt.com/>

[16] Google Cloud. *G Suite: aplicaciones de colaboración y productividad para empresas* [en línia] Alphabet, 1998. [Consulta: 8 març 2019]. Disponible a: <https://gsuite.google.es/intl/es/>

[17] Smartsheet Inc, *Smartsheet* [en línia]. Smartsheet Inc, 2005. [Consulta: 8 març 2019]. Disponible a: <https://es.smartsheet.com/>

[18] JetBrains s.r.o. *JetBrains: Developer Tools for Professionals and Teams* [en línia]. JetBrains s.r.o., 2000. [Consulta: 8 març 2019]. Disponible a: <https://www.jetbrains.com/>

[19] DigitalOcean, LLC. *DigitalOcean - Cloud Computing, Simplicity at Scale* [en línia]. DigitalOcean, LLC, 2011. [Consulta: 8 març 2019]. Disponible a: <https://www.digitalocean.com/>

[20] Departament d'Organització d'Empreses. *Gestión económica* [en línia]. Barcelona: Facultat d'Informàtica de Barcelona (FIB). [Consulta: 11 març 2019]. Disponible a: <https://atenea.upc.edu/pluginfile.php/2669389/mod_folder/content/0/M%C3%B2dul%202.4%20-%20Gesti%C3%B3n%20econ%C3%B3mica.pdf?forcedownload=1>

[21] Michael Page. *Michael Page: Consultora Recursos Humanos y Selección de Personal* [en línia]. 2018. [Consulta: 11 març 2019]. Disponible a: <https://www.michaelpage.es/>

[22] Michael Page. *PageGroup Tecnología: Tendencias del mercado laboral* [en línia]. 2018. [Consulta: 11 març 2019]. Disponible a: <https://www.michaelpage.es/sites/michaelpage.es/files/PG_ER_IT.pdf>

[23] TMB. *Precio tarjeta T-Jove metro bus Barcelona* [en línia]. Barcelona: Transports Metropolitans de Barcelona (TMB). [Consulta: 11 març 2019]. Disponible a: <https://www.tmb.cat/es/tarifas-metro-bus-barcelona/sencillos-e-integrados/t-jove>

[24] Agencia Estatal. *Real Decreto 1777/2004, de 30 de julio, por el que se aprueba el Reglamento del Impuesto sobre Sociedades* [en línia]. Agencia Estatal, Boletín Oficial del Estado (BOE). [Consulta: 11 març 2019]. Disponible a: <https://boe.es/buscar/act.php?id=BOE-A-2004-14600&p=20141128&tn=2>

[25] Standards Coordinating Committee of the Computer Science of the IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. IEEE std 610.12-1990. 28-09-1990 [Consulta l'11 febrer 2019]. Disponible a: <http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf>

[26] Robertson, S.; Robertson, J,. Appendix A Volere Requirements Specification Template. Robertson, S.; Robertson, J,. *Mastering the requirements process: getting requirements right*. Tercera edició. Upper Saddle River : Addison-Wesley, cop. 2013. ISBN 9780321815743 (cart.)

[27] Jakob Nielsen. *10 Usability Heuristics for User Interface Design*[en línia]. Nielsen Norman Group, Abril 1994 [Consulta l'11 febrer 2019]. Disponible a: <https://www.nngroup.com/articles/ten-usability-heuristics/ >

[28] I.Jacobson, G.Booch, J.Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999. ISBN 0201571692.

[29] Departament d'Enginyeria de Serveis i Sistemes d'Informació. *Especificació en UML: Esquema conceptual de les dades* [en línia]. Barcelona: Facultat d'Informàtica de Barcelona (FIB). [Consulta: 21 abril 2019]. Disponible a: https://www.coursehero.com/file/22994353/4-Esquema-conceptual-de-les-dades-4/

[30] Mark Richards. *Software Architecture Patterns* [en línia]. [Consulta: 31 maig 2019]. Disponible a: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

[31] CodeAcademy. *MVC: Model, View, Controller* [en línia]. [Consulta: 31 maig 2019]. Disponible a: https://www.codecademy.com/articles/mvc

[32] ServiceTonic. *¿Qué es API? Definición y ejemplos* [en línia]. [Consulta: 3 juny 2019]. Disponible a: https://www.servicetonic.com/es/service-desk/que-es-api-definicion-y-ejemplos/

[33] Gamma, E., Helm, R., Johnson, R. & Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Academic Internet Publishers, 2011. ISBN 9780201633610

[34] JournalDev. *Builder Design Pattern in Java* [en línia]. 2018 [Consulta: 6 juny 2019]. Disponible a: https://www.journaldev.com/1425/builder-design-pattern-in-java

[35] JournalDev. *Adapter Design Pattern in Java* [en línia]. 2018 [Consulta: 6 juny 2019]. Disponible a: https://www.journaldev.com/1487/adapter-design-pattern-java

[36] JournalDev. *Proxy Design Pattern* [en línia]. 2018 [Consulta: 6 juny 2019]. Disponible a: https://www.journaldev.com/1572/proxy-design-pattern

[37] JournalDev. *Template Method Design Pattern in Java* [en línia]. 2013 [Consulta: 6 juny 2019]. Disponible a: https://www.journaldev.com/1763/template-method-design-pattern-in-java

[38] JournalDev. *Chain of Responsibility Design Pattern in Java* [en línia]. 2013 [Consulta: 6 juny 2019]. Disponible a: https://www.journaldev.com/1617/chain-of-responsibility-design-pattern-in-java

[39] Android Developers. *Cambios en Android 6.0* [en línia]. Alphabet [Consulta: 11 juny 2019]. Disponible a: https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html

[40] Android Developers. *Fragmentos* [en línia]. Alphabet [Consulta: 11 juny 2019]. Disponible a: https://developer.android.com/guide/components/fragments

[41] Material io. *Navigation drawer* [en línia]. Alphabet [Consulta: 12 juny 2019]. Disponible a: https://material.io/design/components/navigation-drawer.html#

[42] Material io. *App bars: bottom* [en línia]. Alphabet [Consulta: 12 juny 2019]. Disponible a: https://material.io/design/components/app-bars-bottom.html#

[43] JournalDev. *State Design Pattern in Java* [en línia]. 2013 [Consulta: 13 juny 2019]. Disponible a: https://www.journaldev.com/1751/state-design-pattern-java

[44] JournalDev. *Observer Design Pattern in Java* [en línia]. 2013 [Consulta: 13 juny 2019]. Disponible a: https://www.journaldev.com/1739/observer-design-pattern-in-java

[45] Nish Anil, olprod. *Design the infrastructure persistence layer* [en línia]. Microsoft, 2018 [Consulta: 15 juny 2019]. Disponible a: https://docs.microsoft.com/es-es/dotnet/standard/microservices-architecture/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design

[46]Facebook. *Stetho* [en línia]. Facebook, 2015 [Consulta: 15 juny 2019]. Disponible a: http://facebook.github.io/stetho/

[47] Wikipedia. *Inyección de dependències* [en línia]. Wikimedia, 2019 [Consulta: 15 juny 2019]. Disponible a: https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias

[48] BBVAOpen4U. *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos* [en línia]. BBVA API_Market, 2016 [Consulta: 16 juny 2019]. Disponible a: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

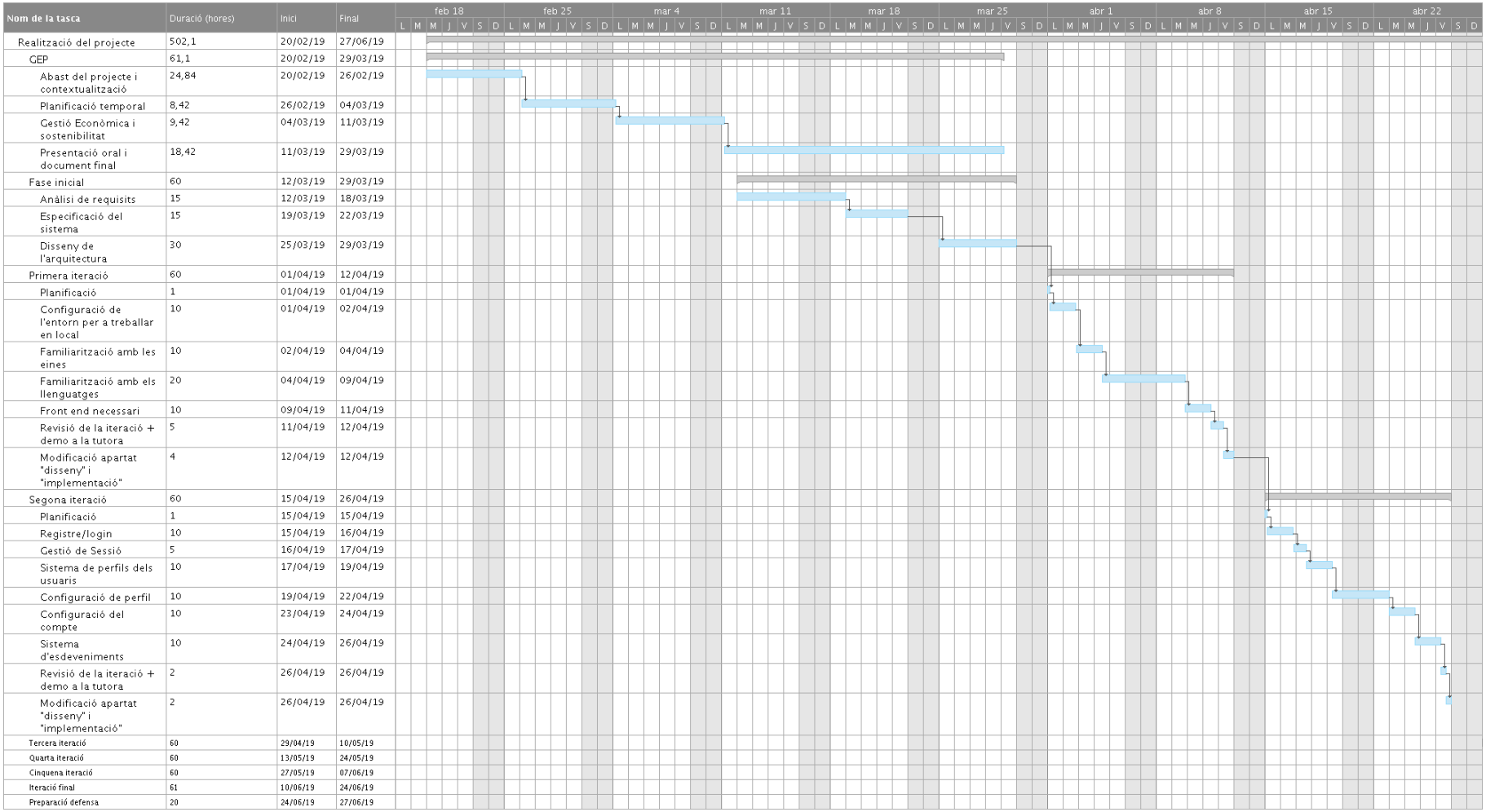
[49] Amazon Web Services. *¿Qué es la integración continua?* [en línia]. Amazon [Consulta: 18 juny 2019]. Disponible a: https://aws.amazon.com/es/devops/continuous-integration/

[50] Sten Pittet. *The different types of software testing* [en línia]. Atlassian [Consulta: 19 juny 2019]. Disponible a: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

[51]Alberto Lacalle. *Pensando en alto: Thinking Aloud* [en línia]. Alberto Lacalle, 2012 [Consulta: 18 juny 2019]. Disponible a: http://albertolacalle.com/hci/thinking-aloud.htm

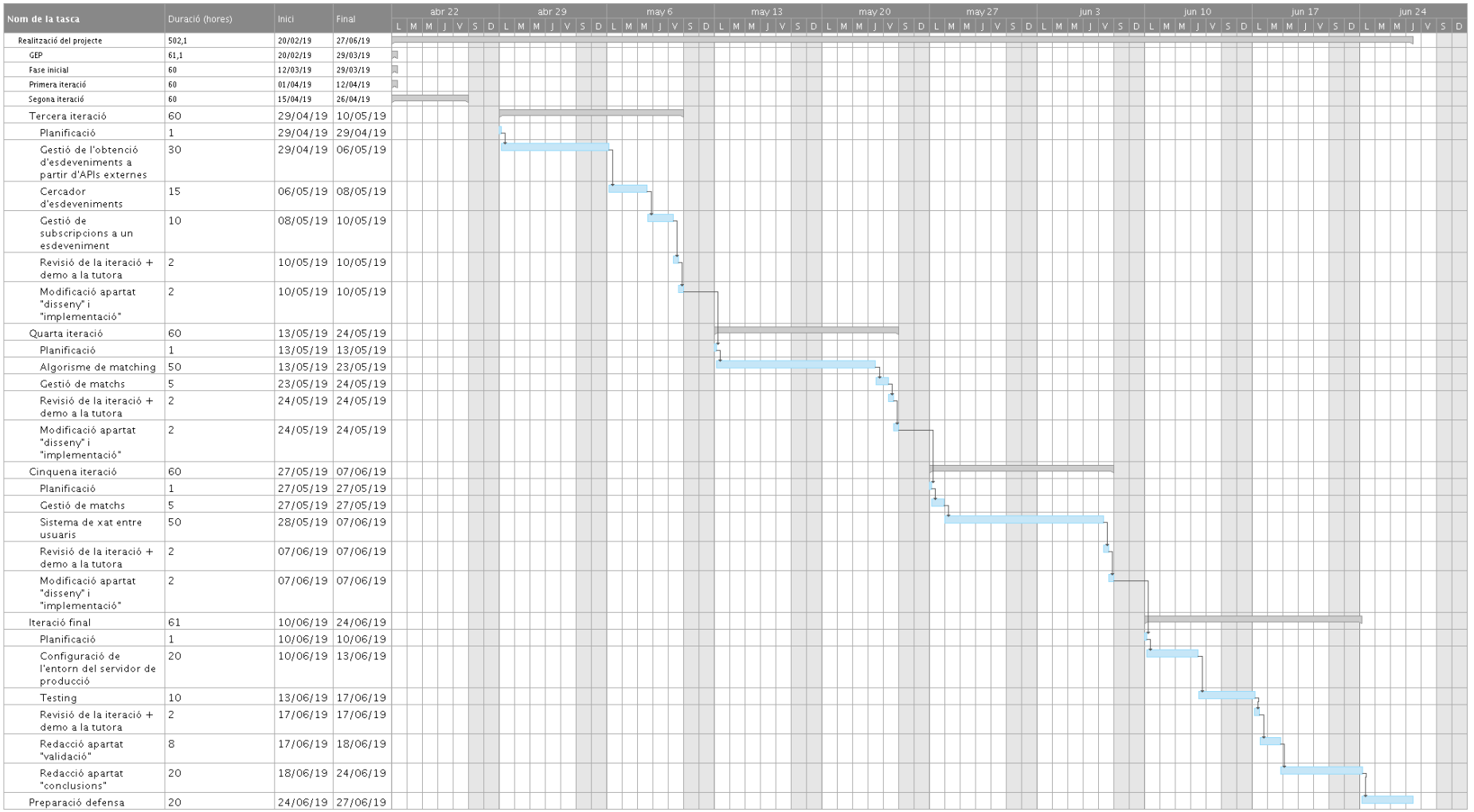
[52] Joan Sánchez. *PartyPal* [en línia]. Barcelona: Google Play Store, 2019 [Consulta: 11 juny 2019]. Disponible a: https://play.google.com/store/apps/details?id=com.JoanSanchez.partypal

Annex 1: Diagrama de *Gantt*



	Nom de la tasca	Duració (hores)	Inici	Final
1	Realització del projecte	502,1	20/02/19	27/06/19
2	GEP	61,1	20/02/19	29/03/19
3	Abast del projecte i contextualització	24,84	20/02/19	26/02/19
4	Planificació temporal	8,42	26/02/19	04/03/19
5	Gestió Econòmica i sostenibilitat	9,42	04/03/19	11/03/19
6	Presentació oral i document final	18,42	11/03/19	29/03/19
7	Fase inicial	60	12/03/19	29/03/19
8	Anàlisi de requisits	15	12/03/19	18/03/19
9	Especificació del sistema	15	19/03/19	22/03/19
10	Disseny de l'arquitectura	30	25/03/19	29/03/19
11	Primera iteració	60	01/04/19	12/04/19
12	Planificació	1	01/04/19	01/04/19
13	Configuració de l'entorn per a treballar en local	10	01/04/19	02/04/19
14	Familiarització amb les eines	10	02/04/19	04/04/19
15	Familiarització amb els llenguatges	20	04/04/19	09/04/19
16	Front end necessari	10	09/04/19	11/04/19
17	Revisió de la iteració + demo a la tutora	5	11/04/19	12/04/19
18	Modificació apartat "disseny" i "implementació"	4	12/04/19	12/04/19
19	Segona iteració	60	15/04/19	26/04/19
20	Planificació	1	15/04/19	15/04/19
21	Registre/login	10	15/04/19	16/04/19
22	Gestió de Sessió	5	16/04/19	17/04/19
23	Sistema de perfils dels usuaris	10	17/04/19	19/04/19
24	Configuració de perfil	10	19/04/19	22/04/19
25	Configuració del compte	10	23/04/19	24/04/19
26	Sistema d'esdeveniments	10	24/04/19	26/04/19
27	Revisió de la iteració + demo a la tutora	2	26/04/19	26/04/19
28	Modificació apartat "disseny" i "implementació"	2	26/04/19	26/04/19
29	Tercera iteració	60	29/04/19	10/05/19
36	Quarta iteració	60	13/05/19	24/05/19
42	Cinquena iteració	60	27/05/19	07/06/19
48	Iteració final	61	10/06/19	24/06/19
55	Preparació defensa	20	24/06/19	27/06/19

Figura 34: Diagrama de Gantt part 1



	Nom de la tasca	Duració (hores)	Inici	Final
1	Realització del projecte	502,1	20/02/19	27/06/19
2	GEP	61,1	20/02/19	29/03/19
7	Fase inicial	60	12/03/19	29/03/19
11	Primera iteració	60	01/04/19	12/04/19
19	Segona iteració	60	15/04/19	26/04/19
29	Tercera iteració	60	29/04/19	10/05/19
30	Planificació	1	29/04/19	29/04/19
31	Gestió de l'obtenció d'esdeveniments a partir d'APIs externes	30	29/04/19	06/05/19
32	Cercador d'esdeveniments	15	06/05/19	08/05/19
33	Gestió de subscripcions a un esdeveniment	10	08/05/19	10/05/19
34	Revisió de la iteració + demo a la tutora	2	10/05/19	10/05/19
35	Modificació apartat "disseny" i "implementació"	2	10/05/19	10/05/19
36	Quarta iteració	60	13/05/19	24/05/19
37	Planificació	1	13/05/19	13/05/19
38	Algorisme de matching	50	13/05/19	23/05/19
39	Gestió de matchs	5	23/05/19	24/05/19
40	Revisió de la iteració + demo a la tutora	2	24/05/19	24/05/19
41	Modificació apartat "disseny" i "implementació"	2	24/05/19	24/05/19
42	Cinquena iteració	60	27/05/19	07/06/19
43	Planificació	1	27/05/19	27/05/19
44	Gestió de matchs	5	27/05/19	27/05/19
45	Sistema de xat entre usuaris	50	28/05/19	07/06/19
46	Revisió de la iteració + demo a la tutora	2	07/06/19	07/06/19
47	Modificació apartat "disseny" i "implementació"	2	07/06/19	07/06/19
48	Iteració final	61	10/06/19	24/06/19
49	Planificació	1	10/06/19	10/06/19
50	Configuració de l'entorn del servidor de producció	20	10/06/19	13/06/19
51	Testing	10	13/06/19	17/06/19
52	Revisió de la iteració + demo a la tutora	2	17/06/19	17/06/19
53	Redacció apartat "validació"	8	17/06/19	18/06/19
54	Redacció apartat "conclusions"	20	18/06/19	24/06/19
55	Preparació defensa	20	24/06/19	27/06/19

Figura 35: Diagrama de Gantt part 2

Annex 2: Índex de figures

Annex 3: Índex de taules